USER GUIDE

Essential Studio

for EJ2 Vue

Version - v24.1.41 | Release Date - December 18, 2023

Syncfusion

Linear Gauge 95
Getting Started with the Vue Linear Gauge Component in Vue 2 95
Prerequisites 95
Dependencies 95
Setting up the Vue 2 project 95
Add Syncfusion Vue packages 96
Adding Syncfusion Vue Linear Gauge component 96
Run the project 97
Module Injection 97
Adding the Linear Gauge Title 98
Axis Range 98
Set Pointer Value 100
Getting Started with the Vue Linear Gauge Component in Vue 3 101
Prerequisites 101
Set up the Vite project 101
Add Syncfusion Vue packages 103
Add Syncfusion Vue component 103
Run the project 106
See also 106
Linear gauge dimensions in Vue Linear gauge component 107
Size for Linear Gauge 107
Axis in Vue Linear gauge component 108
Setting the start value and end value of the axis 108
Line Customization 108
Ticks Customization 109
Labels Customization 111
Orientation 115
Inverted Axis 116
Opposed Axis 117
Multiple Axes 117
Ranges in Vue Linear gauge component 118
Customizing the range 119
Setting the range color for the labels 120
Multiple ranges 120
Gradient Color 121

Pointers in Vue Linear gauge component 124
Types of pointer 125
Bar Pointer 130
Multiple pointers 131
Pointer animation 132
Gradient Color 133
Annotations in Vue Linear gauge component 135
Adding annotation 136
control-container { 136
Customization 137
Multiple annotations 139
Animation in Vue Linear Gauge component 140
User interaction in Vue Linear gauge component 142
Tooltip 142
Pointer Drag 146
Linear gauge print and export in Vue Linear gauge component 147
Print 147
Export 148
Linear gauge appearance in Vue Linear gauge component 151
Customizing the Linear Gauge area 151
Setting up the Linear Gauge title 151
Customizing the Linear Gauge container 152
Fitting Linear Gauge to the control 155
Accessibility in Vue Linear Gauge component 156
WAI-ARIA attributes 156
Screen reading in Linear Gauge 156
Ensuring accessibility 156
See also 157
Internationalization in Vue Linear gauge component 157
Numeric Format 157
Events in Vue Linear gauge component 158
animationComplete 158
annotationRender 158
axisLabelRender 159
beforePrint 160

dragEnd 161
dragMove 162
dragStart 162
gaugeMouseDown 163
gaugeMouseLeave 164
gaugeMouseMove 164
gaugeMouseUp 165
load 166
loaded 167
resized 167
tooltipRender 168
valueChange 169
Methods in Vue Linear gauge component 170
setPointerValue 170
setAnnotationValue 171
refresh 172
ListBox 173
Getting Started with the Vue List box Component in Vue 2 173
Prerequisites 173
Setting up the Vue 2 project 173
Add Syncfusion Vue packages 173
Import Syncfusion CSS styles 174
Add Syncfusion Vue component 174
Binding data source 175
Run the project 177
Getting Started with the Vue ListBox Component in Vue 3 178
Prerequisites 178
Set up the Vite project 178
Add Syncfusion Vue packages 179
Import Syncfusion CSS styles 180
Add Syncfusion Vue component 180
Run the project 183
See also 183
Accessibility in Vue List box component 183
WAI-ARIA attributes 184

Keyboard interaction 184
Ensuring accessibility 185
See also 185
Data binding in Vue List box component 185
Local Data 185
Remote Data 188
Drag and drop in Vue List box component 189
Single listbox 189
Multiple listbox 190
Dual list box in Vue List box component 191
Icons and templates in Vue List box component 193
Icons 193
Templates 196
Selection in Vue List box component 198
Single selection 199
Multiple selection 199
Sorting and grouping in Vue List box component 200
Sorting 200
Grouping 201
Style and appearance in Vue List box component 202
Horizontal ListBox 203
How To 204
Add items in Vue List box component 204
Enable or disable items in Vue List box component 205
Enable scroller in Vue List box component 206
Form submit in Vue List box component 206
Select items in Vue List box component 207
ListView 208
Getting Started with the Vue Listview Component in Vue 2 208
Prerequisites 208
Setting up the Vue 2 project 208
Add Syncfusion Vue packages 209
Import Syncfusion CSS styles 209
Add Syncfusion Vue component 210
Bind data source 210

Run the project 213
See Also 213
Getting Started with the Vue ListView Component in Vue 3 214
Prerequisites 214
Set up the Vite project 214
Add Syncfusion Vue packages 215
Import Syncfusion CSS styles 216
Add Syncfusion Vue component 216
Run the project 218
See also 219
Data binding in Vue Listview component 219
Bind to local data 220
Bind to remote data 222
Grouping in Vue Listview component 223
Customization 224
Check list in Vue Listview component 224
Checkbox Position 225
Nested list in Vue Listview component 226
Customizing templates in Vue Listview component 231
Header template 231
Template 233
Group template 237
Virtualization in Vue Listview component 240
Module injection 240
Getting started 240
Template 241
Conditional rendering 244
Scrolling in Vue Listview component 245
Accessibility in Vue Listview component 248
WAI-ARIA attributes 249
Keyboard interaction 250
Ensuring accessibility 250
See also 250
Style in Vue Listview component 250
Customizing ListView 250

Customizing the list items ... 251
Customizing ListView's header ... 251
Customizing group header of ListView ... 251
Customizing the hover state of ListView control ... 251
Customizing selected item of ListView control ... 252
How To ... 252
Get selected items from listview in Vue Listview component ... 252
Use dynamic templates in listview based on device in Vue Listview component ... 254
Create dual list from listview in Vue Listview component ... 256
Load list items in child list dynamically in Vue Listview component ... 262
Customize listview with dynamic tags in Vue Listview component ... 264
Show items count in group header in Vue Listview component ... 269
Filter list items in the listview in Vue Listview component ... 272
Add and remove list items from listview in Vue Listview component ... 273
Trace all events in listview in Vue Listview component ... 276
Create mobile contact layout from listview in Vue Listview component ... 278
Display spinner until list items are loaded in Vue Listview component ... 282
Hide checkbox in listview in Vue Listview component ... 283
Manipulate listview as grid layout in Vue Listview component ... 289
element .e-list-item { ... 290
Drag and drop list items in Vue Listview component ... 297
Load html content via ajax in Vue Listview component ... 299
Render listview with hyper link navigation in Vue Listview component ... 300
Customize listview as chat window in Vue Listview component ... 302
Maps ... 307
Getting Started with the Vue Maps Component in Vue 2 ... 307
Prerequisites ... 307
Dependencies ... 307
Setting up the Vue 2 project ... 308
Add Syncfusion Vue packages ... 308
Adding Syncfusion Vue Maps component ... 309
Run the project ... 310
Module Injection ... 310
Render shapes from GeoJSON data ... 311
Bind data source to map ... 313

Apply Color Mapping 314
Add Title for Maps 315
Enable Legend 316
Add Data Label 318
Enable Tooltip 319
Getting Started with the Vue Maps Component in Vue 3 320
Prerequisites 320
Set up the Vite project 320
Add Syncfusion Vue packages 322
Add Syncfusion Vue component 322
Run the project 327
See also 328
Populate data in Vue Maps component 328
Geometry types 328
Shape data 329
Data source 329
Data binding 330
Binding complex data source 333
Layers in Vue Maps component 335
Multilayer 335
Sublayer 336
Displaying different layer in the view 337
Rendering custom shapes 338
Providers 338
Map provider in Vue Maps component 338
Bing maps in Vue Maps component 343
Azure maps in Vue Maps component 353
Other maps in Vue Maps component 361
Customization in Vue Maps component 373
Setting the size for Maps 373
Maps title 374
Setting theme 375
Customizing Maps container 376
Customizing Maps area 377
Customizing the shapes 378

Setting color to the shapes from the data source 379
Applying border to individual shapes 380
Projection type 381
Color mapping in Vue Maps component 382
Types of color mapping 383
Multiple colors for a single shape 388
Color for items excluded from color mapping 389
Color mapping for bubbles 390
Data labels in Vue Maps component 391
Adding data labels 391
Customization 394
Label animation 395
Smart labels 396
Intersect action 397
Adding data label as a template 398
Polygon shape in Vue Maps component 399
Markers in Vue Maps component 402
Adding marker 403
Adding marker template 404
Customization 405
Marker shapes 407
Multiple marker groups 408
Customize marker shapes from data source 409
Repositioning the marker using drag and drop 411
Marker zooming 415
Marker Clustering 416
Customization of marker cluster 417
Expanding the marker cluster 419
Tooltip for marker 421
Bubble in Vue Maps component 422
Bubble shapes 423
Customization 424
Setting colors to the bubbles from the data source 425
Setting the range of the bubble size 426
Multiple bubble groups 427

Enable tooltip for bubble 429
Legend in Vue Maps component 430
Modes of legend 431
Positioning of the legend 432
Legend for shapes 434
Enable legend for bubbles 445
Enable legend for markers 446
Annotations in Vue Maps component 448
Annotation 448
Annotation customization 450
Multiple Annotation 453
Navigation line in Vue Maps component 454
Customization 454
Enabling the arrows 455
User interactions in Vue Maps component 457
Zooming 457
Selection 473
Highlight 483
Tooltip 489
Print in Vue Maps component 493
Print 493
Export 495
State persistence in Vue Maps component 501
State Persistence 501
Accessibility in Vue Maps component 502
WAI-ARIA attributes 502
Screen reading in Maps 503
Keyboard Navigation 503
Ensuring accessibility 504
See also 504
Internationalization in Vue Maps component 504
Globalization 504
Numeric Format 504
Localization in Vue Maps component 506
Methods in Vue Maps component 507

Methods 507
getMinMaxLatitudeLongitude 507
How To 509
Annotation in Vue Maps component 509
Custom path in Vue Maps component 511
Drilldown in Vue Maps component 512
Marker type in Vue Maps component 514
Multiple layer in Vue Maps component 516
Zooming in Vue Maps component 517
MaskedTextBox 518
Getting Started with the Vue Maskedtextbox Component in Vue 2 518
Prerequisites 518
Setting up the Vue 2 project 519
Add Syncfusion Vue packages 519
Import Syncfusion CSS styles 520
Add Syncfusion Vue component 520
Adding MaskedTextBox component 521
Set the mask 521
Run the project 523
Model binding 523
See Also 525
Getting Started with the Vue MaskedTextBox Component in Vue 3 525
Prerequisites 525
Set up the Vite project 525
Add Syncfusion Vue packages 527
Import Syncfusion CSS styles 527
Add Syncfusion Vue component 527
Run the project 529
Set the mask 529
Model binding 530
See Also 532
Mask configuration in Vue Maskedtextbox component 532
Standard mask elements 532
Custom mask elements 534
Prompt character 535

Accessibility in Vue Maskedtextbox component 536
WAI-ARIA attributes 537
Ensuring accessibility 537
See also 538
Two way binding in Vue Maskedtextbox component 538
Style appearance in Vue Maskedtextbox component 539
Customizing the appearance of MaskedTextBox wrapper element 539
Customizing the MaskedTextBox element on hovering 539
How To 539
Perform custom validation using form validator in Vue Maskedtextbox component 539
Set cursor position while focus on the input textbox in Vue Maskedtextbox component 541
Display numeric keypad when focus on mobile devices in Vue Maskedtextbox component 542
Customize the ui appearance of the control in Vue Maskedtextbox component 543
Mention 544
Getting Started with the Vue Mention Component in Vue 2 544
Prerequisites 544
Dependencies 544
Setting up the Vue 2 project 545
Add Syncfusion Vue packages 545
Import Syncfusion CSS styles 546
Add Syncfusion Vue component 546
Adding Mention component 547
Binding data source 549
Run the project 553
Display Mention character 553
Getting Started with the Vue Mention Component in Vue 3 555
Prerequisites 555
Set up the Vite project 555
Add Syncfusion Vue packages 557
Import Syncfusion CSS styles 557
Add Syncfusion Vue component 557
Run the project 560
Display Mention character 560
Working with data in Vue Mention component 562
Binding local data 563

Binding remote data 567
Filtering data in Vue Mention component 569
Limit the minimum filter character 569
Change the filter type 571
Allow spacing between search 572
Customize the suggestion item count 573
Sorting in Vue Mention component 575
Template in Vue Mention component 576
Item template 576
Display template 578
No records template 579
Spinner template 580
Localization in Vue Mention component 582
Loading translations 582
Customization in Vue Mention component 583
Show or hide mention character 583
Adding the suffix character after selection 585
Configure the popup list 586
Trigger character 587
Accessibility in Vue Mention component 587
WAI-ARIA attributes 588
Keyboard interaction 589
Ensuring accessibility 590
See also 590
Menu 590
Getting Started with the Vue Menu Component in Vue 2 590
Prerequisites 591
Dependencies 591
Setting up the Vue 2 project 591
Add Syncfusion Vue packages 592
Import Syncfusion CSS styles 592
Add Syncfusion Vue component 593
Initialize Menu 593
Run the project 597
Group menu items with separator 598

See Also 600
Getting Started with the Vue Menu Component in Vue 3 600
Prerequisites 600
Set up the Vite project 600
Add Syncfusion Vue packages 602
Import Syncfusion CSS styles 602
Add Syncfusion Vue component 603
Run the project 607
See also 607
Icons and sub menu items in Vue Menu component 608
Icons 608
Navigation 610
Multilevel nesting 612
Data source binding and custom menu items in Vue Menu component 614
Data binding 614
Custom menu items 619
See Also 623
Use case scenarios in Vue Menu component 623
Scrollable menu 623
Menu in toolbar 627
Hamburger menu 631
Mobile view 635
Accessibility in Vue Menu component 640
WAI-ARIA attributes 641
Keyboard interaction 641
Ensuring accessibility 641
See also 642
Style and appearance in Vue Menu component 642
How To 642
Change orientation in Vue Menu component 642
Customize menu using events in Vue Menu component 643
Customize menu items in Vue Menu component 646
Create mnemonic ui in menuitem in Vue Menu component 651
Change sub menu position in Vue Menu component 654
Set title in Vue Menu component 655

Menu item click in Vue Menu component 658
Message 660
Getting Started with the Vue Message Component in Vue 2 660
Prerequisites 660
Setting up the Vue 2 project 660
Add Syncfusion Vue packages 661
Import Syncfusion CSS styles 661
Add Syncfusion Vue component 661
Run the project 663
Getting Started with the Vue Message Component in Vue 3 664
Prerequisites 664
Set up the Vite project 664
Add Syncfusion Vue packages 665
Import Syncfusion CSS styles 666
Add Syncfusion Vue component 666
Run the project 668
See also 668
Severities in Vue Message component 668
Variants in Vue Message component 669
Icons in Vue Message component 671
No Icon 671
Custom Icon 672
Close Icon 673
Customization in Vue Message component 675
Content Alignment 675
Rounded and Square 676
CSS Message 677
Template in Vue Message component 678
Accessibility in Vue Message component 680
WAI-ARIA attributes 681
Keyboard interaction 681
Ensuring accessibility 681
See also 681
MultiSelect 681
Getting Started with the Vue Multi select Component in Vue 2 681

Prerequisites 681
Setting up the Vue 2 project 682
Add Syncfusion Vue packages 682
Import Syncfusion CSS styles 683
Add Syncfusion Vue component 683
Binding data source 684
Run the project 685
Configure the Popup List 686
See Also 687
Getting Started with the Vue MultiSelect Component in Vue 3 687
Prerequisites 687
Set up the Vite project 687
Add Syncfusion Vue packages 689
Import Syncfusion CSS styles 689
Add Syncfusion Vue component 690
Run the project 691
Configure the Popup List 691
See Also 693
Data binding in Vue Multi select component 693
Binding local data 693
Binding remote data 696
See Also 697
Templates in Vue Multi select component 697
Item template 697
Value template 698
Group template 700
Header template 701
Footer template 703
No records template 704
Action failure template 705
See Also 706
Grouping in Vue Multi select component 706
Customization 707
Grouping with CheckBox 707
See Also 708

Filtering in Vue Multi select component 708
Limit the minimum filter character 710
Change the filter type 711
Case sensitive filtering 712
Diacritics Filtering 713
See Also 714
Custom value in Vue Multi select component 715
Checkbox in Vue Multi select component 715
Select All 716
Selection Limit 717
Selection Reordering 718
See Also 719
Chip customization in Vue Multi select component 719
Localization in Vue Multi select component 721
Loading translations 722
See Also 723
Style in Vue Multi select component 723
Customizing the background color of wrapper element 723
Customizing the appearance of the delimiter wrapper element 723
Customizing the appearance of chips 723
Customizing the dropdown icon’s color 724
Customizing the focus color 724
Customizing the disabled component’s text color 724
Customizing the color of the placeholder text 724
Customizing the placeholder to add mandatory indicator(*) 725
Customizing the float label element's focusing color 725
Customizing the outline theme's focus color 725
Customizing the background color of focus, hover, and active item's 726
Customizing the appearance of pop-up element 726
Customizing the color of the checkbox 726
Accessibility in Vue Multi select component 726
WAI-ARIA attributes 727
Keyboard interaction 728
Ensuring accessibility 730
See also 730

Two way binding in Vue Multi select component 730
How To 731
Icons support in Vue Multi select component 731
NumericTextbox 732
Getting Started with the Vue Numerictextbox Component in Vue 2 732
Prerequisites 732
Setting up the Vue 2 project 732
Add Syncfusion Vue packages 733
Import Syncfusion CSS styles 733
Add Syncfusion Vue component 734
Run the project 735
Range validation 736
Formatting the value 737
Precision of numbers 738
Model binding 740
See Also 741
Getting Started with the Vue NumericTextBox Component in Vue 3 741
Prerequisites 742
Set up the Vite project 742
Add Syncfusion Vue packages 743
Import Syncfusion CSS styles 744
Add Syncfusion Vue component 744
Run the project 745
Range validation 746
Formatting the value 747
Precision of numbers 749
Model binding 750
See Also 752
Formats in Vue Numerictextbox component 752
Standard formats 752
Custom formats 753
Globalization in Vue Numerictextbox component 754
Localization 754
Internationalization 755
Right to Left(RTL) 757

Accessibility in Vue Numerictextbox component 757
WAI-ARIA attributes 758
Keyboard interaction 759
Ensuring accessibility 760
See also 760
Style appearance in Vue Numerictextbox component 761
Customizing the appearance of NumericTextBox wrapper element 761
Customizing the NumericTextBox icons 761
How To 762
Customize the ui appearance of the control in Vue Numerictextbox component 762
Customize the spin buttons up and down arrow in Vue Numerictextbox component 763
Customize the step value and hide spin buttons in Vue Numerictextbox component 764
Maintain trailing zeros in numerictextbox in Vue Numerictextbox component 765
Perform custom validation using form validator in Vue Numerictextbox component 766
Prevent nullable input in numerictextbox in Vue Numerictextbox component 769
Pager 770
Getting started 770
Prerequisites 770
Setup Vue Environment 770
Create a Vue Application 770
Adding Syncfusion Pager package 770
Registering Pager Component 771
Adding CSS Reference 771
Adding Pager component 771
Page Size 773
Page Count 773
Run the application 774
Style and appearance in Vue Pager component 774
Customizing the Pager 774
Accessibility in Vue Pager component 776
WAI-ARIA 776
Keyboard navigation 776
PDF Viewer 776
Getting Started 776
Getting Started with Standalone PDF Viewer component 776

Getting Started with the Vue Pdfviewer Component in Vue 2 781
Getting Started with Syncfusion PDF Viewer Component in Vue 3 785
Prerequisites 786
Set up the Vite project 786
Add Syncfusion Vue packages 787
Import Syncfusion CSS styles 787
Add Syncfusion Vue component 788
Run the project 791
Feature module in Vue Pdfviewer component 792
See also 793
Open PDF files 793
Opening a PDF from URL 793
Opening a PDF from base64 data 795
Opening a PDF from database 797
Saving PDF file 800
Save PDF file to Server 800
Download PDF file as a copy 802
Save PDF file to Database 803
Accessibility in Syncfusion Vue PDF Viewer components 805
WAI-ARIA attributes 806
Keyboard interaction 807
Ensuring accessibility 809
See also 809
Toolbar in Vue Pdfviewer component 809
Show/Hide the default toolbar 810
Show/Hide the default toolbaritem 812
See also 814
Navigation in Vue Pdfviewer component 814
Toolbar page navigation option 814
Bookmark navigation 818
Thumbnail navigation 820
Hyperlink navigation 823
Table of content navigation 823
See also 826
Magnification in Vue Pdfviewer component 826

See also 828
Text search in Vue Pdfviewer component 828
See also 830
Annotation 830
Text markup annotation in Vue Pdfviewer component 830
Shape annotation in Vue Pdfviewer component 850
Stamp annotation in Vue Pdfviewer component 857
Sticky notes annotation in Vue Pdfviewer component 862
Measurement annotation in Vue Pdfviewer component 868
Free text annotation in Vue Pdfviewer component 877
Ink annotation in Vue Pdfviewer component 885
Comments in Vue Pdfviewer component 891
Interaction mode in Vue Pdfviewer component 894
Selection mode 894
Panning Mode 896
See also 898
Form Designer 898
Create form fields programmatically in Vue Pdfviewer component 898
Create form fields with UI interaction in Vue Pdfviewer component 954
Print in Vue Pdfviewer component 961
See also 964
Download in Vue Pdfviewer component 964
See also 966
Globalization in Vue Pdfviewer component 966
How To 972
Toolbar customization in Vue Pdfviewer component 972
pdfviewer { 975
magnificationToolbar { 975
magnificationToolbar.e-toolbar .e-tbar-btn { 975
customToolbar { 975
popup .e-dlg-content { 976
Load documents in Vue Pdfviewer component 984
Unload document in Vue Pdfviewer component 987
Open thumbnail in Vue Pdfviewer component 989
Extract text in Vue Pdfviewer component 990

Import export annotation object in Vue Pdfviewer component 992
Authorization token in Vue Pdfviewer component 994
Delete annotation in Vue Pdfviewer component 995
Convert pixel to point in server side in Vue Pdfviewer component 997
Select annotation in Vue Pdfviewer component 997
Resolve "Unable to find an entry point named FPDFText_GetCharAngle" error 999
How to clear the "Web-service is not listening" to error 1001
Retry Timeout 1003
Load N number of pages on initial loading 1004
Customize toolbar in PDF Viewer component 1007
Troubleshooting 1011
Why Do I Have to Manually Copy Files from node_modules into My App? 1011
Troubleshoot error 'cp' is not recognized as a command 1012
Document Loading Issues in Version 23.1 or Newer 1012
Pivot Table 1013
Getting Started with the Vue Pivotview Component in Vue 2 1013
Prerequisites 1013
Dependencies 1013
Setting up the Vue 2 project 1014
Add Syncfusion Vue packages 1014
Import Syncfusion CSS styles 1015
Add Syncfusion Vue component 1015
Browser compatibility 1016
Assigning sample data to the pivot table 1016
Adding fields to row, column, value and filter axes 1017
Applying formatting to a value field 1018
Module injection 1019
Enable Pivot Field List 1020
Enable Grouping Bar 1021
Exploring Filter Axis 1022
Calculated field 1023
Run the application 1024
Getting Started with the Vue Pivot Table component in Vue 3 1026
Prerequisites 1026
Set up the Vite project 1026

Add Syncfusion Vue packages 1027
Import Syncfusion CSS styles 1028
Add Syncfusion Vue component 1028
Run the project 1033
See also 1041
Data binding in Vue Pivotview component 1041
JSON 1041
CSV 1045
Remote Data Binding 1049
Mapping 1052
Values in row axis 1054
Values at different positions 1055
Show 'no data' items 1056
Show value headers always 1057
Customize empty value cells 1058
Event 1059
See Also 1061
Connecting To Data Source 1061
MySQL in EJ2 Vue Pivotview Component 1061
Microsoft SQL server in EJ2 Vue Pivotview Component 1068
PostgreSQL in EJ2 Vue Pivotview Component 1074
Oracle in EJ2 Vue Pivotview Component 1080
MongoDB in EJ2 Vue Pivotview Component 1087
Elasticsearch in EJ2 Vue Pivotview Component 1094
Snowflake in EJ2 Vue Pivotview Component 1100
Olap in Vue Pivotview component 1107
Getting Started 1107
Run the application 1139
Data Binding 1140
OLAP Cube: Elements 1149
Server side pivot engine in Vue Pivotview component 1151
Quick steps to render the Pivot Table by using the server-side Pivot Engine 1151
Available configurations in Server-side application 1154
Pivot chart in Vue Pivotview component 1174
Data binding 1175

Chart Types 1175
Accumulation Charts 1177
Field List 1186
Grouping Bar 1187
Single Axis 1189
Multiple Axis 1190
Series customization 1195
Axis Customization 1196
Legend customization 1197
User interaction 1199
Export 1203
Print 1204
Drill down in Vue Pivotview component 1205
Drill down and drill up 1205
Drill position 1206
Expand all 1206
Expand all headers for specific fields 1207
Expand all except specific member(s) 1208
Expand specific member(s) 1209
Event 1210
Data Shaping 1214
Aggregation in Vue Pivotview component 1214
Calculated field in Vue Pivotview component 1226
Grouping in Vue Pivotview component 1239
Filtering in Vue Pivotview component 1239
Sorting in Vue Pivotview component 1239
Drill through in Vue Pivotview component 1239
Editing in Vue Pivotview component 1239
Data Formatting 1240
Number formatting in Vue Pivotview component 1240
Conditional formatting in Vue Pivotview component 1240
Report Manipulation 1240
Grouping bar in Vue Pivotview component 1240
Field list in Vue Pivotview component 1240
Defer update in Vue Pivotview component 1270

Performance 1272
Virtual scrolling in Vue Pivotview component 1272
Paging in Vue Pivotview component 1278
State persistence in Vue Pivotview component 1289
Save and Load Pivot Layout 1290
Row and column in Vue Pivotview component 1291
Width and Height 1291
Row Height 1292
Column Width 1293
Reorder 1294
Column Resizing 1295
Text Wrap 1296
Text Align 1296
AutoFit 1298
Grid Lines 1301
Selection 1302
Clip Mode 1308
Cell Template 1309
Events 1313
Summary customization in Vue Pivotview component 1317
Show or hide grand totals 1317
Show grand totals at top or bottom 1318
Show or hide sub-totals 1319
Show or hide sub-totals for specific fields 1320
Show sub-totals at top or bottom 1320
Show or hide totals using toolbar 1322
Hyper link in Vue Pivotview component 1323
Hyperlink for all cells 1324
Hyperlink for row headers 1325
Hyperlink for column headers 1326
Hyperlink for value cells 1327
Hyperlink for summary cells 1328
Condition based hyperlink 1329
Header based hyperlink 1330
Event 1331

See Also 1333
Tool bar in Vue Pivotview component 1333
Built-in Toolbar Options 1333
Show desired chart types in the dropdown menu 1337
Switch the chart to multiple axes 1338
Show or hide legend 1339
Adding custom option to the toolbar 1340
Save and load report as a JSON file 1345
Save and load reports to a SQL database 1346
Events 1373
See Also 1388
Tool tip in Vue Pivotview component 1388
Tooltip Template 1389
Css customization in Vue Pivotview component 1391
Hiding Axis 1391
Text Alignment 1392
Customize header, value and summary cell style 1393
Printing and Exporting 1394
Print in Vue Pivotview component 1394
Excel export in Vue Pivotview component 1397
Pdf export in Vue Pivotview component 1413
Globalization and localization in Vue Pivotview component 1441
Internationalization 1441
Localization 1447
Right-to-left (RTL) 1450
See Also 1452
Accessibility in Vue Pivotview component 1452
WAI-ARIA attributes 1453
Keyboard interaction 1454
Ensuring accessibility 1461
See also 1466
How To 1466
Switching older themes style in Vue Pivotview component 1466
Customize number date and time values in Vue Pivotview component 1468
Customize the icons for pivot grid in Vue Pivotview component 1470

PivotView_PivotFieldList .e-icons.e-toggle-field-list::before { 1470
Configure data grid options on editing mode in Vue Pivotview component 1472
Refresh the field list in Vue Pivotview component 1473
Hide empty headers in Vue Pivotview component 1474
Customizing loading indicator in Vue Pivotview component 1475
Changing the pivotview component minimum height in Vue Pivotview component 1476
Chart based on pivot table selection in Vue Pivotview component 1477
Drill through grid cell edit type in Vue Pivotview component 1481
Show field list when pivot table empty in Vue Pivotview component 1483
Apply custom style to pivot cells in Vue Pivotview component 1484
Show tooltip for row and column headers in Vue Pivotview component 1485
Hide specific columns in Vue Pivotview component 1487
Export table and chart into the same document using toolbar in vue Pivotview component 1488
Add custom aggregation type to the menu in Vue Pivotview component 1489
Convert complex JSON to flat JSON and assign it to the pivot table in Vue Pivotview component 1491
Load desired report from the report list as default in vue Pivotview component 1495
Display string value to pivot table values in vue Pivotview component 1499
Predefined Dialogs 1500
Getting Started with the Vue Predefined dialogs Component in Vue 2 1500
Prerequisites 1500
Dependencies 1501
Setting up the Vue 2 project 1501
Add Syncfusion Vue packages 1502
Import Syncfusion CSS styles 1502
Add Syncfusion Vue component 1502
Creating Vue sample 1503
statusText { 1503
Prerequisites 1508
Set up the Vite project 1508
Add Syncfusion Vue packages 1510
Import Syncfusion CSS styles 1510
Adding Syncfusion Vue3 Predefined Dialogs in the application 1511
Run the project 1512
Draggable in Vue Predefined dialogs component 1512

Alert drag 1512
Confirm drag 1513
Prompt drag 1514
Animation in Vue Predefined dialogs component 1515
Alert animation 1515
Confirm animation 1516
Prompt animation 1516
Position in Vue Predefined dialogs component 1517
Alert position 1517
Confirm position 1518
Prompt position 1519
Dimension in Vue Predefined dialogs component 1520
Alert dimension 1520
Confirm dimension 1520
Prompt dimension 1521
Max-width and max-height 1522
Min-width and min-height 1523
Customization in Vue Predefined dialogs component 1524
Alert action button 1524
Confirm action buttons 1525
Prompt action buttons 1526
Show or hide dialog close button 1527
Alert dialog close button 1527
Confirm dialog close button 1528
Prompt dialog close button 1528
Customize dialog content 1529
ProgressBar 1530
Getting Started with the Vue Progressbar Component in Vue 2 1530
Prerequisites 1530
Dependencies 1530
Setting up the Vue 2 project 1530
Add Syncfusion Vue packages 1531
Import Syncfusion CSS styles 1531
Add Syncfusion Vue component 1532
Run the project 1534

Module Injection 1534
Getting Started with the Vue Progress Bar Component in Vue 3 1535
Prerequisites 1535
Set up the Vite project 1535
Add Syncfusion Vue packages 1536
Add Syncfusion Vue component 1537
Run the project 1539
See also 1540
Types in Vue Progressbar component 1540
Linear 1540
loader { 1542
container { 1542
Circular 1543
loader { 1545
container { 1545
Tooltip in Vue Progressbar component 1545
Tooltip 1545
loader { 1547
container { 1547
Format 1547
loader { 1549
container { 1549
Customization 1549
loader { 1550
container { 1551
States in Vue Progressbar component 1551
Determinate 1551
loader { 1552
container { 1552
Indeterminate 1552
loader { 1553
container { 1554
Buffer 1554
loader { 1555
container { 1555

Customization in Vue Progressbar component 1555
Segments 1555
loader { 1557
container { 1557
Thickness 1557
loader { 1559
container { 1559
Radius 1559
loader { 1561
container { 1561
InnerRadius 1561
loader { 1562
container { 1563
Progress color and track color 1563
loader { 1564
container { 1565
Annotation in Vue Progressbar component 1565
Annotation 1565
loader { 1566
container { 1567
Label 1567
loader { 1568
container { 1569
Animation in Vue Progressbar component 1569
loader { 1570
container { 1570
Range in Vue Progressbar component 1570
loader { 1572
container { 1572
Events in Vue Progressbar component 1573
Value Change 1573
loader { 1574
container { 1575
ProgressCompleted 1575
loader { 1577

container { 1577
Accessibility in Vue Progress bar component 1577
WAI-ARIA attributes 1578
Keyboard interaction 1578
Ensuring accessibility 1578
See also 1579
ProgressButton 1579
Getting Started with the Vue Progress button Component in Vue 2 1579
Prerequisites 1579
Dependencies 1579
Setting up the Vue 2 project 1579
Add Syncfusion Vue packages 1580
Import Syncfusion CSS styles 1580
Add Syncfusion Vue component 1581
Run the project 1582
Enable progress in button 1582
See Also 1583
Getting Started with the Vue ProgressButton Component in Vue 3 1583
Prerequisites 1583
Set up the Vite project 1584
Add Syncfusion Vue packages 1585
Import Syncfusion CSS styles 1585
Add Syncfusion Vue component 1586
Run the project 1588
See also 1588
Spinner and progress in Vue Progress button component 1588
Progress 1589
See Also 1594
Accessibility in Vue Progress button component 1594
WAI-ARIA attributes 1595
Keyboard interaction 1595
Ensuring accessibility 1596
See also 1596
Style and appearance in Vue Progress button component 1596
How To 1596

Change the text content and styles of the progressbutton during progress in Vue Progress button component 1596
Customize progress using cssclass in Vue Progress button component 1597
Hide spinner in Vue Progress button component 1598
Trace events of progress button in Vue Progress button component 1599
QueryBuilder 1600
Getting Started with the Vue Query builder Component in Vue 2 1600
Prerequisites 1600
Dependencies 1600
Setting up the Vue 2 project 1601
Add Syncfusion Vue packages 1601
Import Syncfusion CSS styles 1602
Add Syncfusion Vue component 1602
Run the project 1604
Rendering with rule 1605
Columns in Vue Query builder component 1608
Auto generation 1608
Labels 1610
Operators 1610
Step 1610
Format 1610
Validations 1612
Data binding in Vue Query builder component 1615
Local data 1615
Remote data 1617
Data Manager 1623
Complex Data Binding 1625
Filtering in Vue Query builder component 1628
Templates in Vue Query builder component 1631
Header Template 1631
Column Template 1635
Rule Template 1640
Model binding in Vue Query builder component 1644
Import export in Vue Query builder component 1646
Initial rendering 1646

Post rendering 1648
Exporting 1652
Exporting to JSON 1652
Exporting to SQL 1652
Global local in Vue Query builder component 1655
Style and appearance in Vue Query builder component 1658
Accessibility in Vue Query Builder component 1658
WAI-ARIA attributes 1659
Keyboard interaction 1660
Ensuring accessibility 1660
See also 1660
How To 1660
Display mode in Vue Query builder component 1660
Sort columns in Vue Query builder component 1662
Restrict groups in Vue Query builder component 1664
State persistence in Vue Query builder component 1666
Rtl in Vue Query builder component 1668
Summary view in Vue Query builder component 1670
RadioButton 1672
Getting Started with the Vue Radio button Component in Vue 2 1672
Prerequisites 1672
Dependencies 1672
Setting up the Vue 2 project 1672
Add Syncfusion Vue packages 1673
Import Syncfusion CSS styles 1673
Add Syncfusion Vue component 1673
Run the project 1675
Change the RadioButton state 1675
Getting Started with the Vue RadioButton Component in Vue 3 1677
Prerequisites 1677
Set up the Vite project 1677
Add Syncfusion Vue packages 1678
Import Syncfusion CSS styles 1679
Add Syncfusion Vue component 1679
Run the project 1681

See also 1682
Label and size in Vue Radio button component 1682
Label 1682
Size 1683
See Also 1683
Accessibility in Vue Radio button component 1683
WAI-ARIA attributes 1684
Keyboard interaction 1684
Ensuring accessibility 1685
See also 1685
Style and appearance in Vue Radio button component 1685
How To 1685
Customize radiobutton appearance in Vue Radio button component 1685
Name and value in form submit in Vue Radio button component 1687
Right to left in Vue Radio button component 1688
Set the disabled state in Vue Radio button component 1689
Range Navigator 1690
Getting Started with the Vue Range navigator Component in Vue 2 1690
Prerequisites 1690
Dependencies 1690
Setting up the Vue 2 project 1690
Add Syncfusion Vue packages 1691
Add Syncfusion Vue component 1692
Run the project 1692
Module Injection 1693
Populate Range Navigator with Data 1693
Enable Tooltip 1694
Getting Started with the Vue Range Navigator Component in Vue 3 1695
Prerequisites 1695
Set up the Vite project 1695
Add Syncfusion Vue packages 1697
Add Syncfusion Vue component 1697
Run the project 1700
See also 1701
Selecting range in Vue Range navigator component 1701

Lightweight in Vue Range navigator component 1702
See Also 1702
Series types in Vue Range navigator component 1702
Line 1703
Area 1703
StepLine 1704
Period selector in Vue Range navigator component 1705
Periods 1705
Positioning period selector 1706
Height 1707
Visibility of range navigator 1708
See Also 1709
Labels in Vue Range navigator component 1709
Multilevel labels 1709
Grouping 1710
Smart labels 1711
Label positioning 1712
Labels customization 1712
Grid tick in Vue Range navigator component 1713
Grid line customization 1713
Tick line customization 1714
Customization in Vue Range navigator component 1715
Navigator appearance 1715
Thumb 1716
Border customization 1717
Deferred update 1717
Allow snapping 1718
Animation 1719
See Also 1720
Tool tip in Vue Range navigator component 1720
Customization 1720
Label Format 1721
R t l in Vue Range navigator component 1722
Export print in Vue Range navigator component 1723
Export 1723

Print 1724
Accessibility in Vue Range navigator component 1725
WAI-ARIA attributes 1726
Keyboard interaction 1726
Ensuring accessibility 1726
See also 1726
Range Slider 1726
Getting Started with the Vue Range slider Component in Vue 2 1726
Prerequisites 1726
Setting up the Vue 2 project 1726
Add Syncfusion Vue packages 1727
Import Syncfusion CSS styles 1727
Add Syncfusion Vue component 1728
Run the project 1730
Types 1730
Customization 1732
See Also 1736
Getting Started with the Vue Range Slider Component in Vue 3 1736
Prerequisites 1736
Set up the Vite project 1736
Add Syncfusion Vue packages 1738
Import Syncfusion CSS styles 1738
Add Syncfusion Vue component 1739
Run the project 1740
Types 1740
Customization 1742
See also 1747
Two way binding in Vue Range slider component 1748
Step 1749
Min and Max 1750
Format in Vue Range slider component 1751
Using format API 1752
Using Events 1753
Limits in Vue Range slider component 1754
Default and MinRange Slider limits 1754

Range Slider limits 1755
Handle lock 1756
Style in Vue Range slider component 1757
Customizing the slider track 1757
Customizing the slider handle 1757
Customizing the slider limits 1758
Customizing the slider ticks 1758
Customizing the slider buttons 1758
Accessibility in Vue Range Slider component 1758
WAI-ARIA attributes 1759
Keyboard interaction 1760
Ensuring accessibility 1760
See also 1760
How To 1760
Date range slider in Vue Range slider component 1760
Time range slider in Vue Range slider component 1762
Numeric range slider in Vue Range slider component 1763
Customize slider bar in Vue Range slider component 1765
Customize slider limits in Vue Range slider component 1769
Customize slider ticks label in Vue Range slider component 1771
ticks_slider .e-scale :nth-child(1)::before { 1771
Customize slider thumb in Vue Range slider component 1775
square_slider.e-control.e-slider .e-handle { 1775
circle_slider.e-control.e-slider .e-handle { 1775
oval_slider.e-control.e-slider .e-handle { 1776
Form slider with form validator in Vue Range slider component 1778
Show slider from hidden state in Vue Range slider component 1784
Reversible Range Slider in Vue 1785
Rating 1786
Getting Started with the Vue Rating Component in Vue 2 1786
Dependencies 1786
Setting up the Vue 2 project 1787
Add Syncfusion Vue packages 1787
Import Syncfusion CSS styles 1788
Add Syncfusion Vue component 1788

Run the project 1790
Value 1790
Getting Started with Syncfusion Rating Component in Vue 3 1791
Prerequisites 1791
Set up the Vite project 1791
Add Syncfusion Vue packages 1793
Adding Syncfusion Rating package in the application 1793
Adding CSS reference for Syncfusion Vue Rating component 1793
Adding Syncfusion Vue Rating component in the application 1793
Run the project 1795
See also 1795
Precision modes in Vue Rating component 1795
Labels in Vue Rating component 1796
Label position 1797
Label template 1798
Tool tip in Vue Rating component 1799
Tooltip template 1799
Tooltip customization 1800
Selection in Vue Rating component 1801
Min value 1802
Single selection 1802
Show or hide reset button 1803
Templates in Vue Rating component 1803
Empty (unrated) symbol template 1804
Full (rated) symbol template 1805
Using Emoji icon as rating symbol 1807
Using SVG icon as rating symbol 1808
Using PNG image as rating symbol 1810
Events in Vue Rating component 1811
beforeItemRender 1811
created 1812
onItemHover 1813
valueChanged 1814
Appearance in Vue Rating component 1815
Items count 1815

Disabled ... 1816
Visible ... 1816
Read only ... 1817
CssClass ... 1818
Changing icon using CssClass ... 1820
Accessibility in Vue Rating component ... 1821
Keyboard interaction ... 1821
ARIA attribute ... 1821
Ribbon ... 1822
Getting Started with the Vue Ribbon Component in Vue 2 ... 1822
Prerequisites ... 1822
Dependencies ... 1822
Setting up the Vue 2 project ... 1822
Add Syncfusion Vue packages ... 1823
Import Syncfusion CSS styles ... 1824
Add Syncfusion Vue component ... 1824
Adding Ribbon Tab ... 1824
Adding Ribbon Group ... 1825
Adding Ribbon Items ... 1825
Run the project ... 1832
Getting Started with Syncfusion Ribbon Component in Vue 3 ... 1832
Prerequisites ... 1832
Set up the Vite project ... 1832
Add Syncfusion Vue packages ... 1834
Import Syncfusion CSS styles ... 1834
Add Syncfusion Vue component ... 1835
Run the project ... 1850
See also ... 1851
Modules in Ribbon component ... 1851
Tabs and Groups ... 1852
Adding Tabs ... 1852
Adding Groups ... 1853
Adding Items ... 1853
Ribbon Items ... 1855
Built-in items ... 1855

Custom items 1876
Items display Mode 1878
Enable or disable items 1882
Ribbon Layouts 1884
Classic layout 1884
Simplified layout 1900
Minimized State 1904
File Menu 1905
Visibility 1906
Adding menu items 1907
Open submenu on click 1908
Custom header text 1909
Ribbon Backstage 1911
Adding backstage items 1911
Adding footer items 1915
Adding separator 1919
Back button 1923
Backstage target 1926
Template 1929
Setting width and height 1935
Help Pane 1938
Tooltip 1940
Adding title 1940
Adding content 1942
Adding icon 1944
Customization 1946
Ribbon Resizing 1948
Defining items allowed size 1949
Defining items active size 1950
Events 1950
tabSelected 1950
tabSelecting 1952
ribbonCollapsing 1953
ribbonExpanding 1955
launcherIconClick 1956

Button item events 1958
CheckBox item events 1961
ColorPicker item events 1964
ComboBox item events 1976
DropDown item events 1987
SplitButton item events 1999
GroupButton item events 2012
FileMenu events 2016
Backstage view events 2028
RichText Editor 2030
Getting Started with the Vue Rich text editor Component in Vue 2 2030
Prerequisites 2030
Dependencies 2030
Setting up the Vue 2 project 2030
Add Syncfusion Vue packages 2031
Import Syncfusion CSS styles 2031
Add Syncfusion Vue component 2032
Module injection 2035
Configure the toolbar 2035
Insert images and links 2036
Retrieve the formatted content 2038
Run the project 2038
Add content using valueTemplate property 2040
See Also 2041
Getting Started with the Vue richtexteditor Component in Vue 3 2041
Prerequisites 2041
Set up the Vite project 2041
Add Syncfusion Vue packages 2043
Import Syncfusion CSS styles 2043
Add Syncfusion Vue component 2044
Run the project 2046
Editor modes in Vue Rich text editor component 2046
HTML editor 2047
Markdown editor 2048
See Also 2051

Toolbar in Vue Rich text editor component 2051
Expand Toolbar 2052
Multi-row Toolbar 2053
Floating Toolbar 2054
Toolbar items 2056
Custom tool 2058
Quick inline toolbar 2062
See Also 2065
Styling in Vue Rich text editor component 2065
Font name and size 2065
Custom font and size 2067
Font and Background color 2069
Editor content styles 2070
See Also 2076
Image in Vue Rich text editor component 2076
Upload options 2077
Image delete 2078
Insert from web 2081
Dimension 2081
Caption and Alt Text 2081
Display position 2081
Image with link 2082
Resize 2083
Drag and Drop 2083
See Also 2085
Audio in Vue Rich text editor component 2085
Configure audio tool in the toolbar 2086
Insert audio from the web 2087
Insert audio from local machine 2088
Replacing audio 2094
Delete audio 2095
Display position 2096
Rename audio before inserting 2098
Upload audio with authentication 2101
See Also 2104

Video in Vue Rich text editor component 2104
Configure the video tool in the toolbar 2104
Insert a video from the web 2106
Insert video from local machine 2107
Replacing video 2113
Delete video 2114
Dimension 2115
Display position 2116
Resize video 2117
Rename video before inserting 2118
Upload video with authentication 2122
See Also 2124
Link in Vue Rich text editor component 2124
Insert link 2124
Remove link 2125
Auto-link 2125
Manipulation 2126
See Also 2127
Table in Vue Rich text editor component 2127
Insert table 2127
Quick Toolbar 2129
Table Header 2129
Insert Rows 2129
Insert Columns 2129
Set Color 2129
Delete Table 2130
Vertical Align 2130
Horizontal Align 2130
Table Styles 2130
Table Properties 2131
Table cell merge and split 2131
Emoji Picker in Vue RichTextEditor component 2132
Enabling the toolbar option and custom emojis 2132
Using the shortcut key to open the emoji picker 2137
Navigating and selecting emojis using the keyboard 2137

Markdown in Vue Rich text editor component 2137
Supported Commands 2137
Markdown to HTML 2139
Table 2143
Custom formation 2150
See Also 2154
File browser in Vue Rich text editor component 2154
Required additional dependency 2154
Additional CSS Reference 2154
Format Painter in Vue Rich Text Editor Component | Syncfusion 2156
Enabling the toolbar option for Format Painter 2156
Customization of copy and paste format 2158
Using the shortcut key to copy and paste the format 2161
Form support in Vue Rich text editor component 2161
Render the Rich Text Editor 2161
See Also 2162
Validation in Vue Rich text editor component 2162
Validation rules 2162
Validation message 2164
Globalization in Vue Rich text editor component 2166
Localization 2166
RTL 2179
Miscellaneous in Vue Rich text editor component 2180
Placeholder 2180
Character count 2181
Code view 2183
Undo/Redo manager 2187
Prevention of cross-site scripting (XSS) 2188
Resizable support 2190
Number and Bullet Format Lists 2192
Xhtml validation in Vue Rich text editor component 2194
Attributes 2194
HTML Elements 2194
Inline mode in Vue Rich text editor component 2195
Show on select/click 2195

See Also 2197
Paste cleanup in Vue Rich text editor component 2197
MS Word to HTML 2197
Paste cleanup 2197
Prompt dialog 2198
Paste as plain text 2198
Keep format 2198
Denied tags 2198
Denied attributes 2198
Allowed style properties 2199
Mention integration in Vue Rich text editor component 2201
See Also 2204
Enter key in Vue Rich text editor component 2204
Enter key customization 2204
Shift-Enter key customization 2207
Iframe in Vue Rich text editor component 2209
IFrame attributes 2210
Adding external CSS/Script file 2212
See Also 2213
Third party integration in Vue Rich text editor component 2213
CodeMirror Integration 2214
Exec command in Vue Rich text editor component 2217
Style in Vue Rich text editor component 2219
Customizing the Rich Text Editor's content 2219
Customizing the Rich Text Editor's toolbar 2219
Customizing the Rich Text Editor's character count 2220
Keyboard support in Vue Rich text editor component 2221
HTML formation shortcut key 2221
Markdown formation shortcut key 2223
Custom key config 2226
See Also 2228
Accessibility in Vue Rich text editor component 2228
ARIA attributes 2229
Keyboard interaction 2231
Ensuring accessibility 2233

See also 2233
How To 2233
Add google fonts in Vue Rich text editor component 2233
Change default font family in Vue Rich text editor component 2235
Check image size in Vue Rich text editor component 2236
Customize placeholder style in Vue Rich text editor component 2238
Customize shortcut keys in Vue Rich text editor component 2239
Set the cursor at the specific range in Vue Rich text editor component 2240
Update value in Vue Rich text editor component 2241
Rename images in server in Vue Rich text editor component 2243
File attachment in Vue Rich text editor component 2247
Format code block in Vue Rich text editor component 2251
Schedule 2252
Getting Started with the Vue Schedule Component in Vue 2 2252
Prerequisites 2252
Setting up the Vue 2 project 2252
Add Syncfusion Vue packages 2253
Import Syncfusion CSS styles 2253
Add Syncfusion Vue component 2254
Module injection 2254
Run the project 2255
Populating appointments 2255
Setting date 2258
Setting view 2259
Individual view customization 2260
Getting Started with Syncfusion Scheduler Component in Vue 3 2261
Prerequisites 2261
Set up the Vite project 2261
Adding Syncfusion Vue packages 2262
Import Syncfusion CSS styles 2263
Add Syncfusion Vue component 2263
Run the project 2269
See also 2270
Module injection in Vue Schedule component 2270
Module injection 2271

Scheduler interactions in Vue Schedule component 2271
Appointments in Vue Schedule component 2272
Normal events 2272
Spanned events 2274
All-day events 2274
Expand all day appointments view on initial load 2274
Customize the rendering of the spanned events 2276
Recurring events 2278
Event fields 2286
Adding Custom fields 2290
Customize the order of the overlapping events 2292
Drag and drop appointments 2293
Inline Appointment 2307
Appointment Resizing 2308
Appointment customization 2313
Setting minimum height 2318
Block Dates and Times 2320
Readonly 2322
Make specific events readonly 2323
Restricting event creation on specific time slots 2324
Differentiate the past time events 2326
Appointments occupying entire cell 2327
How to limit maximum number of events to display 2328
Display tooltip for appointments 2329
Appointment filtering 2332
Appointment selection 2334
Deleting multiple appointments 2334
Retrieve event details from the UI of an event 2334
Get the current view appointments 2336
Get the entire appointment collections 2338
Refresh appointments 2340
Data binding in Vue Schedule component 2341
Binding local data 2341
Binding remote data 2342
Loading data via AJAX post 2347

Passing additional parameters to the server 2348
Handling failure actions 2349
Scheduler CRUD actions 2351
Configuring Scheduler with Google API service 2355
Crud actions in Vue Schedule component 2357
Add 2357
Edit 2363
Delete 2375
Drag and drop 2384
Resize 2385
Virtual scrolling in Vue Schedule component 2386
Enabling lazy loading for appointments 2388
See Also 2391
Editor template in Vue Schedule component 2391
Event editor 2391
Customizing event editor using template 2401
Quick popups 2418
More events indicator and popup 2424
Recurrence editor in Vue Schedule component 2434
Customizing the repeat type option in editor 2434
Customizing the End Type Option in Editor 2436
Accessing the recurrence rule string 2437
Set specific value on recurrence editor 2438
Recurrence date generation 2440
Recurrence date generation in server-side 2441
Restrict date generation with specific count 2442
Timezone in Vue Schedule component 2443
Understanding date manipulation in JavaScript 2443
Scheduler with no timezone 2443
Scheduler set to specific timezone 2445
Display events on same time everywhere with no time difference 2446
Set specific timezone for events 2447
Add or remove timezone names to/from the timezone collection 2448
Timezone methods 2450
Views in Vue Schedule component 2452

Setting specific view on scheduler 2452
View specific configuration 2455
Extending view intervals 2473
See Also 2474
Calendar mode in Vue Schedule component 2474
Gregorian Calendar 2474
Islamic Calendar 2475
Resources in Vue Schedule component 2477
Resource fields 2477
Resource data binding 2478
Scheduler with multiple resources 2482
Resource grouping 2483
Customizing parent resource cells 2493
Working with shared events 2495
Simple resource header customization 2497
Customizing resource header with multiple columns 2499
Collapse/Expand child resources in timeline views 2503
Displaying tooltip for resource headers 2504
Choosing among resource colors for appointments 2506
Dynamically add and remove resources 2509
Setting different working days and hours for resources 2511
Hide non-working days when grouped by date 2515
Compact view in mobile 2516
Adaptive UI in desktop 2518
Header rows in Vue Schedule component 2520
Display year and month rows in timeline views 2522
Display week numbers in timeline views 2523
Timeline view displaying dates of a complete year 2524
Customizing the header rows using template 2525
Row auto height in Vue Schedule component 2527
Calendar month view 2527
Timeline views 2529
Timeline views with multiple resources 2530
Appointments occupying entire cell 2531
Header bar in Vue Schedule component 2533

Show or Hide header bar 2533
Customizing header bar using template 2534
Customizing header bar using events 2536
How to display the view options within the header bar popup 2540
Date header customization 2541
Customizing the date range text 2544
Customizing header indent cells 2546
Timescale in Vue Schedule component 2548
Setting different time slot duration 2549
Customizing time cells using template 2550
Hide the timescale 2552
Highlighting current date and time 2553
Working days in Vue Schedule component 2554
Set working days 2555
Hiding weekend days 2556
Show week numbers 2557
Set working hours 2560
Scheduler displaying custom hours 2561
Setting start day of the week 2562
Scroll to specific time and date 2563
See Also 2566
Cell customization in Vue Schedule component 2566
Setting cell dimensions in all views 2566
Check for cell availability 2568
Customizing cells in all the views 2569
Customizing cell header in month view 2573
Customizing the minimum and maximum date values 2574
How to disable multiple cell and row selection in Schedule 2575
State persistence in Vue Schedule component 2575
Exporting in Vue Schedule component 2576
Excel Exporting 2577
Exporting calendar events as ICS file 2589
Import events from other calendars 2592
How to print the Scheduler element 2594
Context menu in Vue Schedule component 2597

Dimensions in Vue Schedule component 2601
Auto Height and Width 2602
Height and Width in pixel 2603
Height and Width in percentage 2604
See Also 2605
Localization in Vue Schedule component 2605
Globalization 2605
Localizing the static Scheduler text 2608
Setting date format 2613
Setting the time format 2614
Displaying Scheduler in RTL mode 2615
See Also 2616
Accessibility in Vue Schedule component 2616
ARIA attributes 2617
Keyboard interaction 2618
Ensuring accessibility 2619
See also 2619
Scheduler styling in Vue Schedule component 2619
Frequently asked questions in Vue Schedule component 2621
Maximum call stack size exceeded 2621
Grouping with empty resources 2622
Not providing e-field in editor template 2622
Missing CSS reference 2622
QuickInfoTemplate at bottom 2623
Not importing culture files while using localization 2624
How To 2625
Add edit and remove events in Vue Schedule component 2625
Set default value for event fields in Vue Schedule component 2630
Open event editor manually in Vue Schedule component 2631
Prevent date navigation in Vue Schedule component 2635
Half yearly view in Vue Schedule component 2636
Set different work hours in Vue Schedule component 2639
Show quick info template in Vue Schedule component 2641
Enable scroll option on all day section in Vue Schedule component 2646
Two way binding in Vue Schedule component 2647

Manual refresh in Vue Schedule component 2650
Sidebar 2652
Getting Started with the Vue Sidebar Component in Vue 2 2652
Prerequisites 2652
Dependencies 2652
Setting up the Vue 2 project 2653
Add Syncfusion Vue packages 2653
Import Syncfusion CSS styles 2654
Add Syncfusion Vue component 2654
Run the project 2656
Enable backdrop 2657
Position 2660
Animate 2668
Close on document click 2671
Enable gestures 2674
See Also 2677
Getting Started with the Vue Sidebar Component in Vue 3 2677
Prerequisites 2677
Set up the Vite project 2677
Add Syncfusion Vue packages 2679
Import Syncfusion CSS styles 2679
Add Syncfusion Vue component 2679
Run the project 2681
Enable backdrop 2682
Position 2684
Animate 2686
Close on document click 2689
Enable gestures 2692
See also 2696
Custom context in Vue Sidebar component 2696
Variations in Vue Sidebar component 2700
Expanding types of sidebar 2700
See Also 2704
Auto close in Vue Sidebar component 2705
Open close 2706

Docking sidebar in Vue Sidebar component 2707
See Also 2711
Style in Vue Sidebar component 2712
Customizing the sidebar 2712
Customizing the sidebar based on the positions 2712
Customizing the sidebar based on the active state 2712
Customizing the sidebar with dock state 2713
Customizing the different types of sidebar 2713
Customizing the backdrop of the sidebar 2714
Customizing the sidebar in the RTL direction 2714
Prevent the animation transition for the Sidebar component 2714
Accessibility in Vue Sidebar component 2715
WAI-ARIA attributes 2716
Keyboard interaction 2716
Ensuring accessibility 2716
See also 2716
How To 2716
Sidebar with listview in Vue Sidebar component 2716
Open close sidebar in Vue Sidebar component 2720
Multiple sidebar in Vue Sidebar component 2722
Sidebar with treeview in Vue Sidebar component 2724
Fixed sidebar in Vue Sidebar component 2733
Sidebar with variation animation in Vue Sidebar component 2747
Signature 2750
Getting Started with the Vue Signature Component in Vue 2 2750
Prerequisites 2750
Dependencies 2750
Setting up the Vue 2 project 2751
Add Syncfusion Vue packages 2751
Add Syncfusion Vue component 2752
Run the project 2754
Getting Started with the Vue Signature Component in Vue 3 2754
Prerequisites 2754
Set up the Vite project 2754
Add Syncfusion Vue packages 2756

Import Syncfusion CSS styles 2756
Add Syncfusion Vue component 2756
Run the project 2758
See also 2759
Customization in Vue Signature component 2759
Stroke Width 2759
Stroke Color 2759
Background Color 2761
Background Image 2762
See Also 2763
Open save in Vue Signature component 2763
Open Signature 2763
Save Signature 2765
Save with Background 2769
Draw in Vue Signature component 2772
Draw 2772
User interaction in Vue Signature component 2774
Undo and Redo 2774
Clear 2774
Disabled 2774
ReadOnly 2774
Accessibility in Vue Signature component 2777
Keyboard interaction 2778
Ensuring accessibility 2778
See also 2778
How To 2778
Integrate toolbar in Vue Signature component 2778
Skeleton 2788
Getting Started with the Vue Skeleton Component in Vue 2 2788
Prerequisites 2788
Dependencies 2788
Setting up the Vue 2 project 2788
Add Syncfusion Vue packages 2789
Import Syncfusion CSS styles 2789
Add Syncfusion Vue component 2790

Run the project 2791
Getting Started with Syncfusion Skeleton Component in Vue 3 2791
Prerequisites 2792
Set up the Vite project 2792
Add Syncfusion Vue packages 2793
Adding CSS reference for Syncfusion Vue Skeleton component 2793
Adding Syncfusion Vue Skeleton component in the application 2794
Skeleton Shapes 2795
Run the project 2797
See also 2797
Shapes in Vue Skeleton component 2797
Circle skeleton shape 2797
Square skeleton shape 2798
Rectangle skeleton shape 2798
Text skeleton shape 2799
Shimmer effect in Vue Skeleton component 2801
Styles in Vue Skeleton component 2802
cssClass 2803
Visible 2803
Accessibility in Vue Skeleton component 2804
ARIA attributes 2804
Smith Chart 2804
Getting Started with the Vue Smithchart Component in Vue 2 2804
Prerequisites 2804
Dependencies 2804
Setting up the Vue 2 project 2805
Add Syncfusion Vue packages 2805
Add Syncfusion Vue component 2806
Run the project 2806
Module Injection 2807
Add series to Smith chart 2808
Add title to Smith chart 2809
Enable marker to the Smith chart 2811
Enable data label to marker 2813
Enable legend for Smith chart 2814

Enable tooltip for the Smith chart series 2816
Getting Started with the Vue Smith Chart Component in Vue 3 2818
Prerequisites 2818
Set up the Vite project 2818
Add Syncfusion Vue packages 2819
Add Syncfusion Vue component 2820
Run the project 2825
See also 2826
Working with data in Vue Smithchart component 2826
Data Binding 2826
Smithchart dimensions in Vue Smithchart component 2828
Size for Container 2828
Size for Smithchart 2829
Title subtitle in Vue Smithchart component 2831
Enable title 2831
Title trim 2832
Smithchart axis in Vue Smithchart component 2833
Labels Customization 2833
Gridlines 2835
Axisline 2836
Smithchart legend in Vue Smithchart component 2838
Position and Alignment 2838
Customization 2842
Toggle Visibility 2846
Smithchart tooltip in Vue Smithchart component 2847
Smithchart marker in Vue Smithchart component 2849
Marker 2849
Datalabels 2851
Smithchart series in Vue Smithchart component 2853
points or datasource 2853
Series customization 2854
Smithchart print in Vue Smithchart component 2855
Print 2855
Export 2857
Accessibility in Vue Smithchart component 2858

WAI-ARIA attributes 2859
Keyboard interaction 2859
Ensuring accessibility 2859
See also 2859
Sparkline 2860
Getting Started with the Vue Sparkline Component in Vue 2 2860
Prerequisites 2860
Dependencies 2860
Setting up the Vue 2 project 2860
Add Syncfusion Vue packages 2861
Add Syncfusion Vue component 2861
Run the project 2862
Module Injection 2862
Bind data source to Sparkline 2863
Change the type of Sparkline 2864
Enable tooltip for Sparkline 2865
Getting Started with the Vue Sparkline Component in Vue 3 2866
Prerequisites 2867
Set up the Vite project 2867
Add Syncfusion Vue packages 2868
Add Syncfusion Vue component 2868
Run the project 2872
See also 2872
Sparkline dimensions in Vue Sparkline component 2872
Size for container 2872
Size for sparkline 2873
Sparkline types in Vue Sparkline component 2875
Axis customization in Vue Sparkline component 2880
Change value type of the sparkline 2880
Change min and max values of axis 2883
Change value of axis 2884
Axis line customization 2885
Special points customization in Vue Sparkline component 2886
Tie point color 2887
Range band in Vue Sparkline component 2888

Range band customization 2888
Multiple range band customization 2889
Marker in Vue Sparkline component 2890
Adding marker to the sparkline 2890
Adding marker to special point 2891
Customizing markers 2892
Data labels in Vue Sparkline component 2892
Enable data label 2892
Customize data label 2893
Format data label text 2894
User interaction in Vue Sparkline component 2895
Tooltip 2895
Track line 2899
Appearance in Vue Sparkline component 2900
Sparkline border 2900
Sparkline padding 2901
Sparkline area customization 2902
Sparkline theme 2903
Localization in Vue Sparkline component 2904
Tooltip format 2904
Rtl 2905
Accessibility in Vue Sparkline component 2905
WAI-ARIA attributes 2906
Keyboard interaction 2906
Ensuring accessibility 2907
See also 2907
SpeedDial 2907
Getting Started with the Vue Speed dial Component in Vue 2 2907
Prerequisites 2907
Dependencies 2907
Setting up the Vue 2 project 2907
Add Syncfusion Vue packages 2908
Import Syncfusion CSS styles 2908
Add Syncfusion Vue component 2909
Run the project 2911

Getting Started with Syncfusion SpeedDial Component in Vue 3 2912
Prerequisites 2912
Set up the Vite project 2912
Add Syncfusion Vue packages 2914
Adding CSS reference for Syncfusion Vue Speed Dial component 2914
Adding Syncfusion Vue Speed Dial component in the application 2914
Run the project 2917
See also 2917
Items in Vue Speed dial component 2918
Icons in Speed Dial items 2918
Animation 2921
Template 2922
Positions in Vue Speed dial component 2922
Opens items on hover 2923
Programmatically show/hide Speed Dial items 2923
Programmatically refresh the position 2925
Display modes in Vue Speed dial component 2926
Linear display mode 2926
Radial display mode (Radial Menu) 2927
Radial menu in Vue Speed dial component 2927
Radial menu direction 2927
Radial menu start and end angle 2928
Offset 2929
Template in Vue Speed dial component 2930
Item template 2931
Popup template 2932
Styles in Vue Speed dial component 2933
SpeedDial button 2933
Disabled 2934
cssClass 2935
Visible 2936
Tooltip 2936
Opens on hover 2937
Modal in Vue Speed dial component 2938
Event in Vue Speed dial component 2938

clicked 2938
created 2939
beforeOpen 2940
onOpen 2941
beforeClose 2942
onClose 2943
beforeItemRender 2944
Accessibility in Vue Speed dial component 2946
Keyboard interaction 2946
ARIA attributes 2947
Spinner 2947
Getting Started with the Vue Spinner Component in Vue 2 2947
Prerequisites 2947
Setting up the Vue 2 project 2947
Add Syncfusion Vue packages 2948
Add Syncfusion Vue component 2948
Run the project 2950
Getting Started with the Vue Spinner Component in Vue 3 2950
Prerequisites 2950
Set up the Vite project 2950
Add Syncfusion Vue packages 2952
Import Syncfusion CSS styles 2952
Adding Syncfusion Vue3 Spinner component in the application 2952
Run the project 2954
Template in Vue Spinner component 2954
Types in Vue Spinner component 2957
Style in Vue Spinner component 2958
Customizing the spinner 2958
SplitButton 2959
Getting Started with the Vue Split button Component in Vue 2 2959
Prerequisites 2959
Dependencies 2959
Setting up the Vue 2 project 2959
Add Syncfusion Vue packages 2960
Import Syncfusion CSS styles 2960

Add Syncfusion Vue component 2961
Run the project 2963
See Also 2964
Getting Started with the Vue SplitButton Component in Vue 3 2964
Prerequisites 2964
Set up the Vite project 2964
Add Syncfusion Vue packages 2965
Import Syncfusion CSS styles 2966
Add Syncfusion Vue component 2966
Run the project 2969
See also 2969
Icons and separator in Vue Split button component 2969
SplitButton icons 2969
Separator 2973
See Also 2976
Popup items in Vue Split button component 2976
Icons 2976
Template 2978
See Also 2983
Accessibility in Vue Split button component 2983
WAI-ARIA attributes 2984
Keyboard interaction 2985
Ensuring accessibility 2985
See also 2985
Style and appearance in Vue Split button component 2985
How To 2985
Create right to left splitbutton in Vue Split button component 2985
Group items in popup in Vue Split button component 2987
Open a dialog on popup item click in Vue Split button component 2988
Set the disabled state in Vue Split button component 2990
Underline a character in a text in Vue Split button component 2992
Splitter 2993
Getting Started with the Vue Splitter Component in Vue 2 2993
Prerequisites 2993
Dependencies 2993

Setting up the Vue 2 project 2993
Add Syncfusion Vue packages 2994
Import Syncfusion CSS styles 2994
Add Syncfusion Vue component 2994
Load content to the pane 2995
Run the project 2996
See Also 2997
Getting Started with the Vue Splitter Component in Vue 3 2997
Prerequisites 2997
Set up the Vite project 2997
Add Syncfusion Vue packages 2999
Import Syncfusion CSS styles 2999
Add Syncfusion Vue component 2999
Run the project 3002
Pane sizing in Vue Splitter component 3002
Auto size panes 3002
Fixed pane 3004
Pane content in Vue Splitter component 3007
HTML Markup 3007
JavaScript UI components 3008
Pane content using selector 3009
Split panes in Vue Splitter component 3011
Horizontal layout 3011
Vertical layout 3012
Separator 3014
Nested Splitter 3016
Add or remove pane 3018
See Also 3020
Expand collapse in Vue Splitter component 3020
Collapsible panes 3020
Programmatically control the expand and collapse action 3022
Specify initial state to panes 3023
See Also 3024
Resizing in Vue Splitter component 3024
Min and Max validation 3024

Prevent resizing ... 3026
Refresh content on resizing ... 3028
Customize the resize grip and cursor ... 3028
See Also ... 3030
Different layouts in Vue Splitter component ... 3030
Code editor style layout ... 3030
Outlook style layout ... 3033
See Also ... 3040
Style in Vue Splitter component ... 3040
Customizing the split bar ... 3040
Customizing the split bar resize handle ... 3040
Customizing the split bar arrows ... 3041
To hide the resize handle in Splitter ... 3042
Globalization in Vue Splitter component ... 3043
RTL ... 3043
See Also ... 3044
Accessibility in Vue Splitter component ... 3044
Keyboard interaction ... 3045
Ensuring accessibility ... 3045
See also ... 3045
Spreadsheet ... 3045
Overview of the Vue Spreadsheet component ... 3045
Key features ... 3045
Getting Started with the Vue Spreadsheet Component in Vue 2 ... 3047
Prerequisites ... 3047
Dependencies ... 3047
Setting up the Vue 2 project ... 3047
Add Syncfusion Vue packages ... 3048
Import Syncfusion CSS styles ... 3048
Add Syncfusion Vue component ... 3049
Run the project ... 3050
See Also ... 3051
Getting Started with the Vue Spreadsheet Component in Vue 3 ... 3051
Prerequisites ... 3051
Set up the Vite project ... 3051

Add Syncfusion Vue packages ... 3052
Import Syncfusion CSS styles ... 3053
Add Syncfusion Vue component ... 3053
Run the project ... 3057
Data binding in Vue Spreadsheet component ... 3058
Local data ... 3058
Remote data ... 3059
Cell data binding ... 3062
Dynamic data binding and Datasource change event ... 3064
Note ... 3066
See Also ... 3066
Open save in Vue Spreadsheet component ... 3066
Open ... 3067
To add custom header during open ... 3069
Supported file formats ... 3071
Save ... 3072
Server Configuration ... 3080
Server Dependencies ... 3081
Note ... 3081
See Also ... 3081
Worksheet in Vue Spreadsheet component ... 3081
Add sheet ... 3081
Delete sheet ... 3083
Rename sheet ... 3083
Headers ... 3083
Gridlines ... 3083
Sheet visibility ... 3085
Note ... 3087
See Also ... 3087
Cell range in Vue Spreadsheet component ... 3087
Wrap text ... 3087
Merge cells ... 3089
Data Validation ... 3092
Clear ... 3096
Note ... 3098

See Also 3098
Editing in Vue Spreadsheet component 3098
Edit cell 3098
Save cell 3098
Cancel editing 3098
Limitations 3100
Note 3100
See Also 3100
Formulas in Vue Spreadsheet component 3100
Usage 3100
Create User Defined Functions / Custom Functions 3101
Formula bar 3106
Named Ranges 3106
Supported Formulas 3109
Formula Error Dialog 3109
Note 3110
See Also 3110
Formatting in Vue Spreadsheet component 3110
Number Formatting 3110
Text and cell formatting 3113
Conditional Formatting 3116
Note 3120
See Also 3120
Freeze pane in Vue Spreadsheet component 3120
Apply freezepanes on UI 3120
FrozenRows 3120
FrozenColumns 3120
Limitations 3122
Note 3122
See Also 3122
Context menu in Vue Spreadsheet component 3122
Context Menu Items in Row Cell 3122
Context Menu Items in Row Header / Column Header 3122
Context Menu Items in Pager 3123
Context Menu Customization 3123

Note 3126
See Also 3126
Template in Vue Spreadsheet component 3126
Note 3131
See Also 3131
Illustrations in Vue Spreadsheet component 3131
Image 3131
Chart 3169
Note 3174
See Also 3175
Rows and columns in Vue Spreadsheet component 3175
Insert 3175
Delete 3178
Limitations of insert and delete 3180
Hide and show 3180
Size 3182
Changing text in column headers 3184
Note 3185
See Also 3185
Filter in Vue Spreadsheet component 3185
Apply filter on UI 3186
Filter by criteria 3186
Filter by cell value 3187
Clear filter 3187
Clear filter on a field 3188
Reapply filter 3188
Known error validations 3188
Limitations 3188
Note 3188
See Also 3188
Sort in Vue Spreadsheet component 3188
Sort by cell value 3189
Data contains header 3190
Case sensitive sort 3191
Sort multiple columns 3191

Custom sort comparer 3193
Known error validations 3193
Limitations 3193
Note 3193
See Also 3194
Link in Vue Spreadsheet component 3194
Insert Link 3194
Edit Hyperlink 3194
Remove Hyperlink 3194
How to change target attribute 3195
Limitations 3197
Note 3197
See Also 3197
Selection in Vue Spreadsheet component 3197
Cell selection 3198
Row selection 3198
Column selection 3199
How to remove selection in the spreadsheet 3201
Limitations 3202
Note 3202
Clipboard in Vue Spreadsheet component 3202
Cut 3202
Copy 3202
Paste 3203
Prevent the paste functionality 3205
Limitations 3206
Note 3206
Scrolling in Vue Spreadsheet component 3207
Finite Scrolling 3207
Virtual Scrolling 3207
Finite scrolling with defined rows and columns 3207
Note 3209
Protect sheet in Vue Spreadsheet component 3209
Protect Sheet 3209
Unprotect Sheet 3211

Unlock the particular cells in the protected sheet....3211
Limitations of Protect Sheet....3213
Protect Workbook....3213
Unprotect Workbook....3215
Limitations of Protect Workbook....3215
Note....3215
See Also....3215
Searching in Vue Spreadsheet component....3215
Find....3216
Replace....3216
Go to....3216
Limitations....3218
Note....3218
Accessibility in Vue Spreadsheet component....3218
WAI-ARIA attributes....3219
Keyboard interaction....3220
Ensuring accessibility....3221
See also....3221
Keyboard shortcuts in Vue Spreadsheet component....3221
Note....3225
See Also....3225
Ribbon in Vue Spreadsheet component....3225
Ribbon Customization....3225
Note....3228
See Also....3228
Global local in Vue Spreadsheet component....3228
Localization....3228
Internationalization....3241
Right to left (RTL)....3244
Note....3245
See Also....3245
Undo redo in Vue Spreadsheet component....3246
Undo....3246
Redo....3246
Update custom actions in UndoRedo collection....3246

Note ... 3248
See Also ... 3248
Use Cases ... 3248
Collaborative editing in Vue Spreadsheet component ... 3248
How To ... 3256
Sort a range by custom list in Vue Spreadsheet component ... 3256
Print in Vue Spreadsheet component ... 3257
Import an excel document using file uploader in ##Platform_Name## Spreadsheet component 3262
Mobile responsiveness in Vue Spreadsheet component ... 3270
Feature list in Vue Spreadsheet component ... 3271
See Also ... 3273
Stepper ... 3273
Getting Started with the Vue Stepper Component in Vue 2 ... 3273
Prerequisites ... 3273
Dependencies ... 3273
Setting up the Vue 2 project ... 3273
Adding Syncfusion Vue packages ... 3274
Import Syncfusion CSS styles ... 3275
Adding Syncfusion Vue component ... 3275
Adding steps ... 3275
Run the project ... 3277
Configure icon and label ... 3278
Getting Started with Syncfusion Stepper Component in Vue 3 ... 3282
Prerequisites ... 3282
Set up the Vite project ... 3282
Add Syncfusion Vue packages ... 3284
Import Syncfusion CSS styles ... 3284
Adding Syncfusion Vue Stepper component ... 3284
Run the project ... 3291
See also ... 3291
Steps in Vue Stepper component ... 3291
Adding steps ... 3291
Optional steps ... 3295
Disabling steps ... 3297
Setting active step ... 3299

Step status ... 3302
Step styling ... 3304
Step validation ... 3307
Step types in Vue Stepper component ... 3307
Default type ... 3307
Label type ... 3309
Indicator type ... 3314
Linear flow in Vue Stepper component ... 3316
Steps validation in Vue Stepper component ... 3318
Template in Vue Stepper component ... 3321
Tooltip in Vue Stepper component ... 3324
Tooltip template ... 3326
Animation in Vue Stepper component ... 3329
Globalization in Vue Stepper component ... 3330
Localization ... 3330
RTL ... 3332
Accessibility in Vue Stepper component ... 3335
Keyboard interaction ... 3335
ARIA attribute ... 3335
Events in Vue Stepper component ... 3335
created ... 3335
stepChanged ... 3336
stepChanging ... 3337
stepClick ... 3338
beforeStepRender ... 3339
Stock Chart ... 3339
Getting Started with the Vue Stock chart Component in Vue 2 ... 3339
Prerequisites ... 3340
Dependencies ... 3340
Setting up the Vue 2 project ... 3340
Add Syncfusion Vue packages ... 3341
Add Syncfusion Vue component ... 3341
Run the project ... 3342
Module Injection ... 3342
Populate Stock Chart with Data ... 3343

Add Stock Chart Title ... 3345
Add Crosshair ... 3346
Add Trackball ... 3347
Getting Started with the Vue Stock Chart Component in Vue 3 ... 3349
Prerequisites ... 3350
Set up the Vite project ... 3350
Add Syncfusion Vue packages ... 3351
Add Syncfusion Vue component ... 3352
Run the project ... 3356
See also ... 3357
Working with data in Vue Stock chart component ... 3357
Local Data ... 3357
See Also ... 3358
Chart dimensions in Vue Stock chart component ... 3358
Size for Container ... 3358
Size for Stock Chart ... 3359
Axis types in Vue Stock chart component ... 3362
DateTime axis ... 3362
DateTimeCategory axis ... 3363
Logarithmic axis ... 3365
See also ... 3367
Axis customization in Vue Stock chart component ... 3367
Axis Crossing ... 3367
Title ... 3368
Tick Lines Customization ... 3370
Grid Lines Customization ... 3371
Multiple Axis ... 3373
Inversed Axis ... 3374
Opposed Position ... 3375
Series types in Vue Stock chart component ... 3377
Line ... 3377
Spline ... 3377
Hilo ... 3377
HiloOpenClose ... 3377
HollowCandle ... 3377

Candle 3377
Trend lines in Vue Stock chart component 3378
Linear 3378
Exponential 3379
Logarithmic 3379
Polynomial 3379
Power 3379
Moving Average 3379
Technical indicators in Vue Stock chart component 3382
Accumulation Distribution 3382
Average True Range (ATR) 3382
Exponential Moving Average (EMA) 3382
Momentum 3382
Moving Average Convergence Divergence (MACD) 3382
Relative Strength Index (RSI) 3382
Simple Moving Average (SMA) 3382
Stochastic 3383
Triangular Moving Average (TMA) 3383
Bollinger Band 3383
Legend in Vue Stock chart component 3384
Position and Alignment 3384
Customization 3389
Collapsing Legend Item 3393
Legend Title 3395
Tool tip in Vue Stock chart component 3397
Default tooltip 3397
Format the tooltip 3398
Position the tooltip 3400
Customize the appearance of the tooltip 3401
Cross hair in Vue Stock chart component 3402
Tooltip for axis 3404
Customization 3405
Period selector in Vue Stock chart component 3406
Periods 3407
Visibility of period selector 3408

Range selector in Vue Stock chart component 3410
Export print in Vue Stock chart component 3411
Disable Export and print 3412
Appearance in Vue Stock chart component 3414
Stock Chart Title 3414
Title Customization 3415
Stock Chart Theme 3416
See Also 3418
Stock events in Vue Stock chart component 3418
Accessibility in Vue Stock chart component 3426
WAI-ARIA attributes 3427
Keyboard interaction 3427
Ensuring accessibility 3428
See also 3428
Switch 3428
Getting Started with the Vue Switch Component in Vue 2 3428
Prerequisites 3428
Dependencies 3428
Setting up the Vue 2 project 3428
Add Syncfusion Vue packages 3429
Import Syncfusion CSS styles 3429
Add Syncfusion Vue component 3430
Run the project 3431
Set text on Switch 3431
Getting Started with the Vue Switch Component in Vue 3 3431
Prerequisites 3432
Set up the Vite project 3432
Add Syncfusion Vue packages 3433
Import Syncfusion CSS styles 3433
Add Syncfusion Vue component 3434
Run the project 3436
See also 3436
Accessibility in Vue Switch component 3436
WAI-ARIA attributes 3437
Keyboard interaction 3437

Ensuring accessibility 3437
See also 3438
Two way binding in Vue Switch component 3438
Style and appearance in Vue Switch component 3439
How To 3439
Change size in Vue Switch component 3439
Enable ripple for switch label in Vue Switch component 3440
Enable rtl in Vue Switch component 3441
Set disabled state in Vue Switch component 3441
Submit name and value in form in Vue Switch component 3442
Change switch state using toggle method in Vue Switch component 3443
Tab 3444
Getting Started with the Vue Tab Component in Vue 2 3444
Prerequisites 3444
Dependencies 3444
Setting up the Vue 2 project 3444
Add Syncfusion Vue packages 3445
Import Syncfusion CSS styles 3445
Add Syncfusion Vue component 3446
Run the project 3448
Initialize the Tab using HTML elements 3449
Initialize the Tab using V-slot Template 3450
See Also 3451
Getting Started with Syncfusion Vue Tab Component in Vue 3 3452
Prerequisites 3452
Setup the Vite project 3452
Adding Syncfusion Vue packages 3453
Import Syncfusion CSS styles 3454
Adding Syncfusion Vue Tab component in the application 3454
Run the project 3459
See also 3459
Adaptive in Vue Tab component 3459
Scrollable 3459
Popup 3463
See Also 3465

Header in Vue Tab component 3465
Styles 3465
Icon positions 3468
See Also 3473
Localization in Vue Tab component 3473
Loading translations 3473
Orientation in Vue Tab component 3474
Drag and drop in Vue Tab component 3478
Drag and drop item between tabs 3480
Drag and drop items to external source 3484
Drag and drop items from external source 3487
Accessibility in Vue Tab component 3490
ARIA attributes 3491
Keyboard interaction 3492
Ensuring accessibility 3492
See also 3492
Style in Vue Tab component 3492
Customizing Tab 3493
Customizing the Tab items 3493
Customizing Tab's header 3493
Customizing Tab's header icon 3493
Customizing Tab's content 3494
Customizing the hover state of Tab control 3494
Customizing selected item of Tab control 3494
How To 3495
Load content through post in Vue Tab component 3495
Prevent content swipe selection in Vue Tab component 3496
Customize selected tab styles in Vue Tab component 3498
Customize tab scroll step in Vue Tab component 3501
Create wizard using tab in Vue Tab component 3503
Load tab with data source in Vue Tab component 3513
Add nested tabs in Vue Tab component 3514
Set state persistence of the tab component in Vue Tab component 3518
Set custom animation in Vue Tab component 3519
Load tab items dynamically in Vue Tab component 3521

Create collapsible tabs in Vue Tab component 3524
Render other components in tab using template in Vue Tab component 3527
Adding dynamic items with content reuse in Vue Tab component 3529
Customize tab content height in Vue Tab component 3531
Reorder active tab in Vue Tab component 3533
Tab selection in Vue Tab component 3536
Tab key navigation in Vue Tab component 3539
Textbox 3541
Getting Started with the Vue Textbox Component in Vue 2 3541
Prerequisites 3541
Dependencies 3541
Setting up the Vue 2 project 3541
Add Syncfusion Vue packages 3542
Import Syncfusion CSS styles 3542
Adding TextBox to the application 3543
Adding icons to the TextBox 3543
Run the project 3543
Floating label 3547
See Also 3548
Getting Started with the Vue TextBox Component in Vue 3 3548
Prerequisites 3548
Set up the Vite project 3548
Add Syncfusion Vue packages 3550
Import Syncfusion CSS styles 3550
Add Syncfusion Vue component 3550
Adding icons to the TextBox 3552
Run the project 3552
Floating label 3553
See Also 3554
Groups in Vue Textbox component 3554
With icon and floating label 3555
With clear button and floating label 3558
Floating label without required attribute 3559
Multi-line input with floating label 3561
See Also 3562

Sizing in Vue Textbox component 3563
Validation in Vue Textbox component 3565
Adding mandatory asterisk to placeholder and float label 3567
Multiline in Vue Textbox component 3568
Create multiline textbox 3568
Implementing floating label 3569
Auto resizing 3570
Disable resizing 3571
Limit the text length 3573
Count characters 3574
Accessibility in Vue Textbox component 3575
WAI-ARIA attributes 3576
Ensuring accessibility 3576
See also 3576
Two way binding in Vue Textbox component 3576
Style appearance in Vue Textbox component 3577
Customizing the appearance of TextBox wrapper element 3577
Customizing the TextBox placeholder 3578
How To 3578
Set the rounded corner in Vue Textbox component 3578
Set the disabled state in Vue Textbox component 3581
Set the read only textbox in Vue Textbox component 3582
Add textbox programmatically in Vue Textbox component 3583
Add floating label to textbox programmatically in Vue Textbox component 3586
Change the floating label color of the textbox in Vue Textbox component 3589
Change the color of the textbox based on its value in Vue Textbox component 3591
Add floating label to read only textbox in Vue Textbox component 3593
Customize the textbox background color and text color in Vue Textbox component 3594
TimePicker 3596
Getting Started with the Vue Timepicker Component in Vue 2 3596
Prerequisites 3596
Dependencies 3596
Setting up the Vue 2 project 3597
Add Syncfusion Vue packages 3597
Import Syncfusion CSS styles 3598

Add Syncfusion Vue component ... 3598
Run the project ... 3600
Setting the value, min, and max time ... 3601
Setting the time format ... 3602
See Also ... 3603
Getting Started with the Vue TimePicker Component in Vue 3 ... 3603
Prerequisites ... 3604
Set up the Vite project ... 3604
Add Syncfusion Vue packages ... 3605
Import Syncfusion CSS styles ... 3606
Add Syncfusion Vue component ... 3606
Run the project ... 3608
Setting the value, min, and max time ... 3608
Setting the time format ... 3610
See Also ... 3611
Time range in Vue Timepicker component ... 3612
Time masking in Vue Timepicker component ... 3613
Configure Mask Placeholder ... 3614
Globalization in Vue Timepicker component ... 3616
Right-To-Left ... 3618
Strict mode in Vue Timepicker component ... 3619
Enable Strict Mode ... 3619
Disable Strict Mode ... 3620
Accessibility in Vue Timepicker component ... 3621
WAI-ARIA attributes ... 3622
Keyboard Interaction ... 3622
Ensuring accessibility ... 3624
See also ... 3624
Two way binding in Vue Timepicker component ... 3624
Style appearance in Vue Timepicker component ... 3625
Customizing the appearance of TimePicker wrapper element ... 3625
Customizing the TimePicker icon element ... 3625
Customizing the TimePicker popup ... 3626
Customizing the TimePicker popup content ... 3626
Full screen mode support in mobiles and tablets ... 3626

How To 3629
Css customization in Vue Timepicker component 3629
Client side validation using form validator in Vue Timepicker component 3631
Toast 3632
Getting Started with the Vue Toast Component in Vue 2 3632
Prerequisites 3632
Dependencies 3632
Setting up the Vue 2 project 3632
Add Syncfusion Vue packages 3633
Import Syncfusion CSS styles 3633
Add Syncfusion Vue component 3634
Run the project 3635
See Also 3636
Getting Started with the Vue Toast Component in Vue 3 3636
Prerequisites 3636
Set up the Vite project 3636
Add Syncfusion Vue packages 3637
Import Syncfusion CSS styles 3638
Add Syncfusion Vue component 3638
Run the project 3640
Config in Vue Toast component 3641
Title and content template 3641
Specifying custom target 3642
Close button 3642
Progress bar 3642
Newest on top 3642
Width and height 3644
See Also 3647
Position in Vue Toast component 3647
Predefined 3647
Custom 3647
See Also 3652
Action buttons in Vue Toast component 3652
See Also 3654
Timeout in Vue Toast component 3654

Static toast 3655
See Also 3657
Template in Vue Toast component 3657
See Also 3659
Animation in Vue Toast component 3659
Toast services in Vue Toast component 3662
Show Toast with predefined types 3662
Show Toast with ToastModel 3663
Style in Vue Toast component 3665
Customizing the toast title 3665
Customizing the toast content 3665
Customizing the toast icon 3665
Customizing the toast background 3666
Accessibility in Vue Toast component 3666
WAI-ARIA attributes 3667
Ensuring accessibility 3668
See also 3668
How To 3668
Prevent duplicate toast display in Vue Toast component 3668
Restrict the maximum toast to show in Vue Toast component 3669
Customize progress bar theme and sizing in Vue Toast component 3671
Play an audio before open the toast in Vue Toast component 3673
Show different types of toast in Vue Toast component 3674
Show multiple toasts in various positions in Vue Toast component 3675
Close the toast with click tap in Vue Toast component 3677
Add dynamic template in Vue Toast component 3678
Prevent toast close with mobile swipe in Vue Toast component 3679
Toolbar 3680
Getting Started with the Vue Toolbar Component in Vue 2 3680
Prerequisites 3680
Dependencies 3680
Setting up the Vue 2 project 3681
Add Syncfusion Vue packages 3681
Import Syncfusion CSS styles 3682
Add Syncfusion Vue component 3682

Run the project 3684
Initialize the Toolbar using HTML elements 3684
See Also 3685
Getting Started with Syncfusion Vue Toolbar Component in Vue 3 3685
Prerequisites 3685
Setup the Vite project 3685
Adding Syncfusion Vue Toolbar package in the application 3687
Import Syncfusion CSS styles 3687
Adding Syncfusion Vue Toolbar component in the application 3687
Run the project 3705
See also 3705
Item configuration in Vue Toolbar component 3705
Button 3705
Separator 3705
See Also 3709
Template configuration in Vue Toolbar component 3709
Integrate menu component 3709
Responsive mode in Vue Toolbar component 3711
Scrollable 3711
Popup 3718
Accessibility in Vue Toolbar component 3725
ARIA attributes 3726
Keyboard interaction 3726
Ensuring accessibility 3727
See also 3727
Style in Vue Toolbar component 3727
Customizing Toolbar 3727
Customizing the Toolbar items 3727
Customizing Toolbar's item icon 3728
Customizing the hover state of Toolbar control 3728
Customizing selected item of Toolbar control 3728
How To 3729
Set command customization in Vue Toolbar component 3729
Set tool tip to the commands in Vue Toolbar component 3731
Set item wise custom template in Vue Toolbar component 3732

Add toggle button in Vue Toolbar component 3733
Customize toolbar scroll step in Vue Toolbar component 3738
Add link to toolbar item in Vue Toolbar component 3747
Render other components in toolbar using template in Vue Toolbar component 3749
Tooltip 3750
Getting Started with the Vue Tooltip Component in Vue 2 3750
Prerequisites 3750
Setting up the Vue 2 project 3750
Add Syncfusion Vue packages 3751
Import Syncfusion CSS styles 3751
Add Syncfusion Vue component 3752
Run the project 3753
See Also 3754
Getting Started with the Vue Tooltip Component in Vue 3 3754
Prerequisites 3754
Set up the Vite project 3754
Add Syncfusion Vue packages 3755
Import Syncfusion CSS styles 3756
Add Syncfusion Vue component 3756
Run the project 3758
Tooltip Positioning 3758
See also 3761
Setting dimension in Vue Tooltip component 3761
Height and width 3761
Scroll mode 3761
Content in Vue Tooltip component 3763
Template content 3763
Dynamic content via Fetch 3764
Position in Vue Tooltip component 3766
Tip pointer positioning 3768
Dynamic positioning 3768
Mouse trailing 3770
Setting offset values 3771
Open mode in Vue Tooltip component 3772
Custom open mode 3774

Sticky mode 3775
Open/Close Tooltip with delay 3776
Animation in Vue Tooltip component 3777
Animation effects 3778
Animating via open/close methods 3778
Apply transition 3779
Customization in Vue Tooltip component 3781
Tip pointer customization 3781
Tooltip customization 3783
Style in Vue Tooltip component 3785
Customizing the tooltip 3785
Customizing the tooltip popup 3785
Customizing the tooltip content 3785
Customizing the tooltip arrow tip 3786
Customizing the tooltip inner tip 3786
Customizing the tooltip outer tip 3787
Accessibility in Vue Tooltip component 3788
WAI-ARIA attributes 3789
Keyboard interaction 3789
Ensuring accessibility 3789
See also 3790
How To 3790
Show tooltip on disabled elements and disable tooltip in Vue Tooltip component 3790
Dynamic tooltip content with html elements in Vue Tooltip component 3791
Custom tooltip with dynamic html in Vue Tooltip component 3792
Tooltip open or display modes in Vue Tooltip component 3794
Fancy tooltip customization in Vue Tooltip component 3797
Display tooltip on svg and canvas elements in Vue Tooltip component 3803
Dynamic tooltip content in Vue Tooltip component 3806
Create and show tooltip on multiple targets in Vue Tooltip component 3812
TreeGrid 3814
Getting Started with the Vue Treegrid Component in Vue 2 3814
Prerequisites 3814
Setting up the Vue 2 project 3815
Add Syncfusion Vue packages 3815

Import Syncfusion CSS styles 3816
Add Syncfusion Vue component 3816
Defining Row Data 3817
Defining Columns 3818
Module injection 3819
Enable Paging 3819
Enable Sorting 3822
Enable Filtering 3825
Run the application 3828
Getting Started with the Vue Tree Grid Component in Vue 3 3831
Prerequisites 3831
Set up the Vite project 3831
Add Syncfusion Vue packages 3833
Import Syncfusion CSS styles 3833
Add Syncfusion Vue component 3833
Run the project 3839
See also 3840
Module in Vue Treegrid component 3840
Data Binding 3841
Data binding in Vue Treegrid component 3841
Local data in Vue Treegrid component 3845
Remote data in Vue Treegrid component 3847
Immutable mode in Vue Treegrid component 3876
Limitations 3880
Columns 3880
Columns in Vue Treegrid component 3880
Headers in Vue Treegrid component 3889
Column template in Vue Treegrid component 3892
Complex data binding in Vue Treegrid component 3894
Column reorder in Vue Treegrid component 3895
Column resizing in Vue Treegrid component 3897
Column menu in Vue Treegrid component 3899
Responsive columns in Vue Treegrid component 3903
Row in Vue Treegrid component 3904
Row height in Vue Treegrid component 3905

Row template in Vue Treegrid component 3907
Detail template in Vue Treegrid component 3911
Row drag and drop in Vue Treegrid component 3912
Indent in Vue Treegrid component 3916
Cell 3918
Content in Vue Treegrid component 3918
Auto wrap in Vue Treegrid component 3918
Clip mode in Vue Treegrid component 3919
Editing 3920
Edit in Vue Treegrid component 3920
Edit types in Vue Treegrid component 3928
Cell editing in Vue Treegrid component 3930
Row editing in Vue Treegrid component 3931
Dialog editing in Vue Treegrid component 3932
Batch editing in Vue Treegrid component 3933
Command column editing in Vue Treegrid component 3934
Validation in Vue Treegrid component 3936
Persisting data in server in Vue Treegrid component 3938
Sorting in Vue Treegrid component 3945
Initial sort 3947
Sorting events 3947
Touch interaction 3948
Filtering 3949
Filtering in Vue Treegrid component 3949
Filter bar in Vue Treegrid component 3954
Filter menu in Vue Treegrid component 3955
Excel like filter in Vue Treegrid component 3957
Searching in Vue Treegrid component 3959
Initial search 3960
Search operators 3961
Search by external button 3961
Search specific columns 3962
Paging in Vue Treegrid component 3963
Page Size Mode 3964
Pager with Page Size Dropdown 3964

How to render Pager at the Top of the TreeGrid 3965
Scrolling in Vue Treegrid component 3966
Set width and height 3966
Responsive with parent container 3967
Scroll to selected row 3968
Frozen in Vue Treegrid component 3969
Frozen rows and columns 3969
Virtual scroll in Vue Treegrid component 3972
Row virtualization 3972
Column virtualization 3973
Limitations for virtualization 3975
Infinite scroll in Vue Treegrid component 3976
InitialBlocks 3977
Cache Mode 3978
Limitations for Infinite Scrolling 3979
Selection 3979
Selection in Vue Treegrid component 3979
Cell selection in Vue Treegrid component 3981
Row selection in Vue Treegrid component 3982
Check box selection in Vue Treegrid component 3983
Aggregates 3985
Aggregates in Vue Treegrid component 3985
Footer aggregate in Vue Treegrid component 3987
Print in Vue Treegrid component 3989
Page setup 3990
Print using an external button 3990
Print visible Page 3991
Print large number of columns 3992
Show or Hide columns while Printing 3992
Limitations of Printing Large Data 3994
Adaptive in Vue Treegrid component 3994
Render adaptive dialogs 3994
Loading animation in Vue Treegrid component 3996
State Persistence 3997
State persistence in Vue Treegrid component 3997

Get or set local storage value in Vue Treegrid component 3997
Tool Bar 3998
Tool bar in Vue Treegrid component 3998
Tool bar items in Vue Treegrid component 3999
Pdf Export 4003
Pdf export in Vue Treegrid component 4003
Pdf export options in Vue Treegrid component 4005
Pdf cell style customization in Vue Treegrid component 4012
Adding header and footer in Vue Treegrid component 4015
Exporting tree grid in server in Vue Tree Grid component 4019
Excel Export 4023
Excel export in Vue Treegrid component 4023
Excel export options in Vue Treegrid component 4025
Excel cell style customization in Vue Treegrid component 4030
Adding header and footer in Vue Treegrid component 4032
Exporting tree grid in Server in Vue Tree Grid Component 4034
Global local in Vue Treegrid component 4039
Localization 4039
Internationalization 4044
Right to left (RTL) 4046
Accessibility in Vue Treegrid component 4047
WAI-ARIA attributes 4048
Keyboard interaction 4049
Ensuring accessibility 4050
See also 4050
Clipboard in Vue Treegrid component 4050
Copy to clipboard by external buttons 4051
Copy Hierarchy Modes 4052
AutoFill 4054
Paste 4055
Context menu in Vue Treegrid component 4056
Custom context menu items 4058
Enable and disable context menu items dynamically 4059
Treegrid styling in Vue Treegrid component 4061
How To 4062

Refresh the data source in Vue Treegrid component 4062
Enable disable grid and its actions in Vue Treegrid component 4064
Change header text dynamically in Vue Treegrid component 4066
Customize column styles in Vue Treegrid component 4068
Custom tool tip for columns in Vue Treegrid component 4069
Change orientation of header text in Vue Treegrid component 4071
Customize the icon for column menu in Vue Treegrid component 4072
Customize the edit dialog in Vue Treegrid component 4074
Cascading drop down list with treegrid editing in Vue Treegrid component 4075
Provide custom data source and enabling filtering to drop down list in Vue Treegrid component 4077
Restrict decimal points with treegrid editing in Vue Treegrid component 4079
Exporting filtered data in Vue Treegrid component 4080
Exporting selected data in Vue Treegrid component 4081
Show spinner on exporting in Vue Treegrid component 4082
Passing parameter to server exporting in Vue Treegrid component 4084
Customize pager drop down in Vue Treegrid component 4085
Add parameter on filtering in Vue Treegrid component 4086
Select treegrid rows based on certain condition in Vue Treegrid component 4087
Get row cell index in Vue Treegrid component 4088
Display foreign key column in treegrid in Vue Treegrid component 4089
TreeMap 4091
Getting Started with the Vue TreeMap Component in Vue 2 4091
Prerequisites 4091
Dependencies 4091
Setting up the Vue 2 project 4091
Add Syncfusion Vue packages 4092
Adding Syncfusion Vue TreeMap component 4092
Run the project 4094
Module Injection 4095
Render tree map 4095
Apply color mapping 4096
Enable legend 4098
Getting Started with the Vue TreeMap Component in Vue 3 4100
Prerequisites 4100

Set up the Vite project 4100
Add Syncfusion Vue packages 4101
Add Syncfusion Vue component 4102
Run the project 4107
See also 4108
Data binding in Vue Treemap component 4108
Populate data 4108
Layout in Vue Treemap component 4109
Types of layout 4109
Leaf item in Vue Treemap component 4112
Leaf label 4112
Item gap 4115
Levels in Vue Treemap component 4116
Group path 4116
Group gap 4117
Header format and Alignment 4119
Header height and style 4120
Header template and position 4121
Color mapping in Vue Treemap component 4123
Range color mapping 4123
Equal color mapping 4124
Desaturation color mapping 4125
Palette color mapping 4126
Desaturation with multiple colors 4127
Color for items excluded from color mapping 4128
Bind the colors to the items from data source 4129
Data label in Vue Treemap component 4130
Format 4130
Template 4130
InterSectAction 4131
Legend in Vue Treemap component 4132
Position and alignment 4132
Legend mode 4135
Legend size 4137
Legend for items excluded from color mapping 4139

Hide desired legend items 4140
Hide legend items based data source value 4142
Bind legend item text from data source 4143
Hide duplicate legend items 4144
Legend Responsiveness 4145
Drilldown in Vue Treemap component 4145
Perform drill-down action 4146
On-demand data loading 4147
Breadcrumb support 4148
Tooltip in Vue Treemap component 4150
Default tooltip 4150
Format tooltip 4151
Tooltip template 4151
Selection and highlight in Vue Treemap component 4153
Selection 4153
Highlight 4154
Print and export in Vue Treemap component 4156
Print 4156
Export 4157
Accessibility in Vue Treemap component 4160
WAI-ARIA attributes 4161
Screen reading in TreeMap 4161
Ensuring accessibility 4161
See also 4161
Internationalization in Vue Treemap component 4161
Globalization 4162
Right-to-left rendering 4163
Legend with Rtl support 4163
Tooltip with Rtl support 4164
Treemap Item Rendering Direction 4165
How To 4168
Drilldown in Vue Treemap component 4168
Drilldown with label in Vue Treemap component 4172
TreeView 4175
Getting Started with the Vue TreeView Component in Vue 2 4175

Prerequisites 4175
Setting up the Vue 2 project 4175
Adding syncfusion packages 4176
Import Syncfusion CSS styles 4176
Add Syncfusion Vue component 4177
Run the application 4181
Getting Started with the Vue TreeView Component in Vue 3 4181
Prerequisites 4181
Set up the Vite project 4181
Add Syncfusion Vue packages 4183
Import Syncfusion CSS styles 4183
Add Syncfusion Vue component 4183
Run the project 4187
CheckBox 4188
See also 4190
Data binding in Vue Treeview component 4190
Local data 4191
Remote data 4194
Check box in Vue Treeview component 4196
Checked nodes 4197
See Also 4199
Node editing in Vue Treeview component 4199
See Also 4202
Multiple selection in Vue Treeview component 4202
Selected nodes 4203
Drag and drop in Vue Treeview component 4205
Multiple-node drag and drop 4207
See Also 4208
Template in Vue Treeview component 4208
See Also 4210
Accessibility in Vue Treeview component 4210
WAI-ARIA attributes 4211
Keyboard interaction 4212
Ensuring accessibility 4213
See also 4213

How To 4213
Customize the expand and collapse icons in Vue Treeview component 4213
Process the tree node operations using context menu in Vue Treeview component 4216
Check uncheck the checkbox on clicking the tree node text in Vue Treeview component 4220
Validate the text when renaming the tree node in Vue Treeview component 4221
Customize the tree nodes based on levels in Vue Treeview component 4223
Restrict the drag and drop for particular tree nodes in Vue Treeview component 4226
Uploader 4229
Getting Started with the Vue Uploader Component in Vue 2 4229
Prerequisites 4229
Dependencies 4229
Setting up the Vue 2 project 4229
Add Syncfusion Vue packages 4230
Import Syncfusion CSS styles 4230
Add Syncfusion Vue component 4231
Run the project 4233
Adding drop area 4233
Configure asynchronous settings 4236
Handle success and failed upload 4237
See Also 4238
Getting Started with the Vue Uploader Component in Vue 3 4238
Prerequisites 4239
Set up the Vite project 4239
Add Syncfusion Vue packages 4240
Import Syncfusion CSS styles 4240
Add Syncfusion Vue component 4241
Run the project 4243
Adding drop area 4243
Configure asynchronous settings 4246
Handle success and failed upload 4247
See Also 4249
Async in Vue Uploader component 4249
Multiple file upload 4249
Single file upload 4250
Save Action 4251

Remove Action 4253
Auto Upload 4255
Sequential Upload 4256
Preload Files 4256
Adding additional HTTP headers with upload action 4257
See Also 4259
Chunk upload in Vue Uploader component 4259
Additional configurations 4260
Resumable upload 4262
Cancel upload 4263
Server-Side configurations 4264
File source in Vue Uploader component 4268
Paste to upload 4268
Directory upload 4270
Drag and drop 4272
See Also 4275
Validation in Vue Uploader component 4275
File type 4275
File size 4276
Maximum files count 4277
Duplicate files 4278
See Also 4280
Form support in Vue Uploader component 4280
Template in Vue Uploader component 4285
File list template 4285
Custom template 4295
See Also 4301
Localization in Vue Uploader component 4302
Wai aria accessibility in Vue Uploader component 4304
Keyboard interaction 4305
Ensuring accessibility 4306
See also 4306
Style appearance in Vue Uploader component 4306
Customizing the appearance of File Upload wrapper element 4306
Customizing the File Upload browse button 4306

Customizing the File Upload content 4307
Customizing the uploaded file container in File Upload 4307
See Also 4307
How To 4307
Hide default drop area in Vue Uploader component 4307
Preview images before uploading in Vue Uploader component 4309
Achieve invisible upload in Vue Uploader component 4309
Customize progressbar in Vue Uploader component 4311
Sort the selected files in Vue Uploader component 4312
Get the total size of selected files in Vue Uploader component 4314
Customize button with html element in Vue Uploader component 4315
Add confirm dialog to remove the files in Vue Uploader component 4317
Add additional data on upload in Vue Uploader component 4318
container { 4319
loader { 4320
Validate image on drop in Vue Uploader component 4320
Determine whether uploader has file input in Vue Uploader component 4322
Achieve file upload programmatically in Vue Uploader component 4326
Check file size before uploading it in Vue Uploader component 4328
Check the mime type of file before upload it in Vue Uploader component 4329
Trigger click event of input file from external button in Vue Uploader component 4330
Open and edit the uploaded files in Vue Uploader component 4331
container { 4333
loader { 4333
Resize images before uploading it to the server in Vue Uploader component 4335
dropTarget { 4341
dropElement { 4342
dropTarget progress { 4345
dropTarget span a { 4345
Convert image into binary format after uploading in Vue Uploader component 4345

Linear Gauge

Getting Started with the Vue Linear Gauge Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Linear Gauge component

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The list of dependencies required to use the Linear Gauge component in your application is given below:

```
|-- @syncfusion/ej2-vue-lineargauge
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-vue-base
|-- @syncfusion/ej2-svg-base
|-- @syncfusion/ej2-lineargauge
```

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

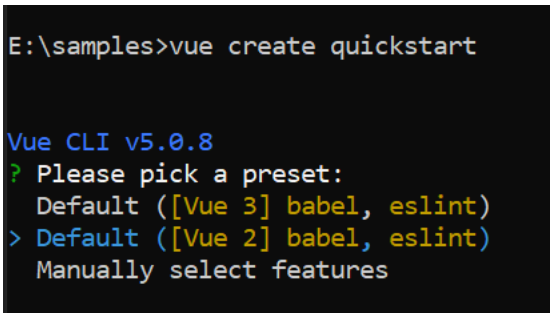
```bash
npm install -g @vue/cli
vue create quickstart
cd quickstart
npm run serve
```

or

```bash
yarn global add @vue/cli
vue create quickstart
cd quickstart
yarn run serve
```

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.

Copyright © 2001 -2024 Syncfusion Inc.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Linear Gauge component as an example. Install the @syncfusion/ej2-vue-lineargauge package by running the following command:

`bash

npm install @syncfusion/ej2-vue-lineargauge --save

`

or

`bash

yarn add @syncfusion/ej2-vue-lineargauge

`

Adding Syncfusion Vue Linear Gauge component

Follow the below steps to add the Vue Linear Gauge component:

1\. First, import and register the Linear Gauge component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { LinearGaugeComponent } from '@syncfusion/ej2-vue-lineargauge';
export default {
components: {
'ejs-lineargauge': LinearGaugeComponent
}
}
</script>
```

2\. In the template section, define the Linear Gauge component.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-lineargauge></ejs-lineargauge>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

~/SRC/APP.VUE

```
<template>
    <div>
        <div class="content-wrapper">
            <div align='center'>
                <ejs-lineargauge ></ejs-lineargauge>
            </div>
        </div>
  </div>
</template>
<script>
import { LinearGaugeComponent } from '@syncfusion/ej2-vue-lineargauge';
export default {
    components: {
        'ejs-lineargauge': LinearGaugeComponent
    }
}
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs34" %}

Module Injection

LinearGauge component is segregated into the individual feature-wise modules. In order to use a particular feature, inject its feature module using the provide: {}. Please find the feature module name and description as follows.

- Annotation - Inject this module to use the annotation features.
- GaugeTooltip - Inject this module to use the tooltip features.

Copyright © 2001 -2024 Syncfusion Inc.

Now import the above mentioned modules from linear gauge package and inject it into the LinearGauge component using provide: {} .

`ts

import { LinearGaugeComponent, Annotations, GaugeTooltip } from "@syncfusion/ej2-vue-lineargauge";

provide: {

lineargauge: [Annotations, GaugeTooltip]

},

`

Adding the Linear Gauge Title

The title can be added to the Linear Gauge component using the title property in the Linear Gauge.

~/SRC/APP.VUE

```
<template>
    <div>
        <div class="content-wrapper">
            <ejs-lineargauge style='display:block' :title ='title' ></ejs-
lineargauge>
        </div>
    </div>
</template>
<script>
import { LinearGaugeComponent } from "@syncfusion/ej2-vue-lineargauge";
export default {
    components: {
        'ejs-lineargauge': LinearGaugeComponent
    },
    data: function () {
        return {
            title: 'linear gauge',
        }
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs35" %}

Axis Range

The range of the axis can be set using the minimum and maximum properties in the Linear Gauge.

~/SRC/APP.VUE

```
<template>
   <div>
    <div class="content-wrapper">
        <ejs-lineargauge >
```

```
            <e-axes>
                <e-axis minimum = 0 maximum = 200 ></e-axis>
            </e-axes>
        </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
"@syncfusion/ej2-vue-lineargauge";
export default {
    components: {
        'ejs-lineargauge': LinearGaugeComponent,
        'e-axes': AxesDirective,
        'e-axis': AxisDirective
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs36" %}

Customizing the axis labels

To denote the axis labels with temperature units, add the °C as suffix to each label. This can be achieved by setting the **{value}°C** to the format property in the labelStyle object of the axis. Here, **{value}** acts as a placeholder for each axis label.

To change the pointer value from the default value of the gauge, set the value property in pointers object of the axis.

~/SRC/APP.VUE

```
<template>
    <div>
        <div class="content-wrapper">
            <div align='center'>
                <ejs-lineargauge>
                    <e-axes>
                        <e-axis minimum = 0 maximum = 200  :ranges='ranges'>
                            <e-pointers>
                                <e-pointer value=140 ></e-pointer>
                            </e-pointers>
                        </e-axis>
                    </e-axes>
                </ejs-lineargauge>
            </div>
        </div>
    </div>
</template>
<script>
import { LinearGaugeComponent, AxesDirective, AxisDirective } from
"@syncfusion/ej2-vue-lineargauge";
```

```
export default {
    components: {
        'ejs-lineargauge': LinearGaugeComponent,
        'e-axes': AxesDirective,
        'e-axis': AxisDirective
    },
    data: function () {
        return {
            labelStyle: {
            format: '{value}°C'
        },
            ranges: [{
            start: 0,
            end: 80,
            startWidth:15,
            endWidth:15
        }, {
            start: 80,
            end: 120,
            startWidth:15,
            endWidth:15
        }, {
            start: 120,
            end: 140,
            startWidth:15,
            endWidth:15
        }, {
            start: 140,
            end: 200,
            startWidth:15,
            endWidth:15
        }]
        }
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs37" %}

Set Pointer Value

The pointer value is changed in the below sample using the value property in pointers object of the axis.

~/SRC/APP.VUE

```
<template>
   <div>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge>
                <e-axes>
                    <e-axis minimum=0 maximum=200>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                        <e-pointers>
                            <e-pointer value=40 color='green'></e-pointer>
                        </e-pointers>
                    </e-axis>
                </e-axes>
            </ejs-lineargauge>
        </div>
    </div>
  </div>
</template>
<script>
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointerDirective, PointersDirective } from "@syncfusion/ej2-vue-
lineargauge";
export default {
    components: {
        'ejs-lineargauge': LinearGaugeComponent,
        'e-axes': AxesDirective,
        'e-axis': AxisDirective,
        'e-pointer': PointerDirective,
        'e-pointers': PointersDirective
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs38" %}

Getting Started with the Vue Linear Gauge Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Linear Gauge component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Linear Gauge component as an example. To use the Vue Linear Gauge component in the project, the @syncfusion/ej2-vue-lineargauge package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-lineargauge --save

`

or

`bash

yarn add @syncfusion/ej2-vue-lineargauge

`

Add Syncfusion Vue component

Follow the below steps to add the Vue Linear Gauge component using Composition API or Options API:

1.First, import and register the Linear Gauge component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { LinearGaugeComponent as EjsLineargauge, AxesDirective as EAxes,
AxisDirective as EAxis, PointersDirective as EPointers, PointerDirective as
EPointer, RangesDirective as ERanges, RangeDirective as ERange } from
'@syncfusion/ej2-vue-lineargauge';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, RangesDirective, RangeDirective } from
'@syncfusion/ej2-vue-lineargauge';
//Component registration
export default {
name: "App",
components: {
'ejs-lineargauge' : LinearGaugeComponent,
'e-axes' : AxesDirective,
'e-axis' : AxisDirective,
'e-pointers': PointersDirective,
'e-pointer' : PointerDirective,
'e-ranges' : RangesDirective,
'e-range' : RangeDirective
}
}
</script>
```

2.In the template section, define the Linear Gauge component with the axis, pointer and range property definitions.

~/SRC/APP.VUE

```
<template>
<ejs-lineargauge :title ='title' orientation='Horizontal'>
<e-axes>
<e-axis minimum=0 maximum=200>
<e-pointers>
<e-pointer value=140 color='red'></e-pointer>
</e-pointers>
<e-ranges>
<e-range start=0 end=80 startWidth=15 endWidth=15></e-range>
<e-range start=80 end=120 startWidth=15 endWidth=15></e-range>
<e-range start=120 end=140 startWidth=15 endWidth=15></e-range>
<e-range start=140 end=200 startWidth=15 endWidth=15></e-range>
</e-ranges>
</e-axis>
</e-axes>
</ejs-lineargauge>
</template>
```

3.In the script section, declare the values for the properties defined in the template section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const title = "Linear Gauge";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
title: 'Linear Gauge'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-lineargauge :title ='title' orientation='Horizontal'>
<e-axes>
<e-axis minimum=0 maximum=200>
<e-pointers>
<e-pointer value=140 color='red'></e-pointer>
</e-pointers>
<e-ranges>
<e-range start=0 end=80 startWidth=15 endWidth=15></e-range>
<e-range start=80 end=120 startWidth=15 endWidth=15></e-range>
<e-range start=120 end=140 startWidth=15 endWidth=15></e-range>
<e-range start=140 end=200 startWidth=15 endWidth=15></e-range>
</e-ranges>
</e-axis>
</e-axes>
</ejs-lineargauge>
</template>
<script setup>
import { LinearGaugeComponent as EjsLineargauge, AxesDirective as EAxes,
AxisDirective as EAxis, PointersDirective as EPointers, PointerDirective as
EPointer, RangesDirective as ERanges, RangeDirective as ERange } from
'@syncfusion/ej2-vue-lineargauge';
const title = "Linear Gauge";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-lineargauge :title ='title' orientation='Horizontal'>
<e-axes>
<e-axis minimum=0 maximum=200>
<e-pointers>
<e-pointer value=140 color='red'></e-pointer>
</e-pointers>
<e-ranges>
<e-range start=0 end=80 startWidth=15 endWidth=15></e-range>
<e-range start=80 end=120 startWidth=15 endWidth=15></e-range>
<e-range start=120 end=140 startWidth=15 endWidth=15></e-range>
<e-range start=140 end=200 startWidth=15 endWidth=15></e-range>
</e-ranges>
</e-axis>
</e-axes>
</ejs-lineargauge>
</template>
<script>
import { LinearGaugeComponent, AxesDirective, AxisDirective,
PointersDirective, PointerDirective, RangesDirective, RangeDirective } from
'@syncfusion/ej2-vue-lineargauge';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
// Component registration
export default {
name: "App",
// Declaring component and its directives
components: {
'ejs-lineargauge' : LinearGaugeComponent,
'e-axes' : AxesDirective,
'e-axis' : AxisDirective,
'e-pointers': PointersDirective,
'e-pointer' : PointerDirective,
'e-ranges' : RangesDirective,
'e-range' : RangeDirective
},
// Bound properties declarations
data() {
return {
title: 'Linear Gauge'
};
}
};
</script>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

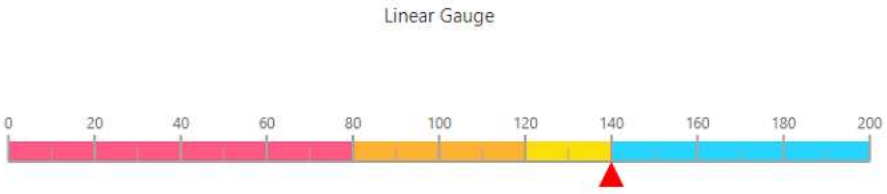
or

`bash

yarn run dev

`

The output will appear as follows:



Sample: vue3-lineargauge-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript

Copyright © 2001 -2024 Syncfusion Inc.

- Getting Started with Vue UI Components using Options API and TypeScript

Linear gauge dimensions in Vue Linear gauge component

Size for Linear Gauge

The height and width of the Linear Gauge can be set using the width and height properties in ejs-lineargauge.

In Pixel

The size of the Linear Gauge can be set in pixel as demonstrated below.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge  width= '650px' height= '200px'></ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from '@syncfusion/ej2-vue-lineargauge';
Vue.use(LinearGaugePlugin);
export default { }
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs46" %}

In Percentage

By setting value in percentage, Linear Gauge receives its dimension matching to its parent. For example, when the height is set as **50%**, Linear Gauge renders to half of the parent height. The Linear Gauge will be responsive when the width is set as **100%**.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge  width= '100%' height= '50%'></ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from '@syncfusion/ej2-vue-lineargauge';
Vue.use(LinearGaugePlugin);
export default { }
</script>
<style>
#content-wrapper {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs47" %}

Note: When the component's size is not specified, the height will be **450px** and the width will be the same as the parent element's width.

Axis in Vue Linear gauge component

Axis is used to indicate the numeric values in the linear scale. The Linear Gauge component can have any number of axes. The sub-elements of an axis are line, ticks, labels, ranges, and pointers.

Setting the start value and end value of the axis

The start value and end value for the Linear Gauge can be set using the minimum and maximum properties in the e-axis respectively. By default, the start value of the axis is **0** and the end value of the axis is **100**.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge >
        <e-axes>
          <e-axis :minimum=20 :maximum=200></e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
    }
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs5" %}

Line Customization

The following properties in the line can be used to customize the axis line in the Linear Gauge.

- height - To set the length of the axis line.

Copyright © 2001 -2024 Syncfusion Inc.

- width - To set the thickness of the axis line.
- color - To set the color of the axis line.
- offset - To render the axis line with the specified distance from the Linear Gauge.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge >
        <e-axes>
          <e-axis :line='line'></e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
      line: {
        height: 150,
        width: 2,
        color: '#4286f4',
        offset: 2
      }
    }
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs6" %}

Ticks Customization

Ticks are used to specify the interval in the axis. Ticks are of two types, major ticks and minor ticks. The following properties in the majorTicks and minorTicks can be used to customize the major ticks and minor ticks respectively.

- height - To set the length of the major and minor ticks in pixel values.
- color - To set the color of the major and minor ticks of the Linear Gauge.
- width - To set the thickness of the major and minor ticks in pixel values.
- interval - To set the interval for the major ticks and minor ticks in the Linear Gauge.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge >
        <e-axes>
          <e-axis minimum= 20  maximum= 140 :majorTicks='majorTicks'
:minorTicks='minorTicks'>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
      majorTicks: {
        interval: 20,
        color: "Orange"
      },
      minorTicks: {
        interval: 5,
        color: 'red'
      }
    }
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs7" %}

Positioning the ticks

The minor and major ticks can be positioned by using the offset and position properties. The offset is used to render the ticks with the specified distance from the axis. By default, the offset value is **0**. The possible values of the position property are **Inside**, **Outside**, **Cross**, and **Auto**. By default, the ticks will be placed inside the axis.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge>
        <e-axes>
          <e-axis minimum= 20  maximum= 140 :majorTicks='majorTicks'
:minorTicks='minorTicks'>
          </e-axis>
```

```
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
      majorTicks: {
        interval: 20,
        color: "Orange",
        position: "Outside"
      },
      minorTicks: {
        interval: 5,
        color: 'red',
        position: "Cross"
      }
    }
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs8" %}

Labels Customization

The style of the labels can be customized using the following properties in the font property in labelStyle.

- color - To set the color of the axis label.
- fontFamily - To set the font family of the axis label.
- fontStyle - To set the font style of the axis label.
- fontWeight - To set the font weight of the axis label.
- opacity - To set the opacity of the axis label.
- size - To set the size of the axis label.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge >
        <e-axes>
          <e-axis :labelStyle='labelStyle2'>
          </e-axis>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
      labelStyle2: {
        font: {
          color: 'red'
        }
      }
    }
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs9" %}

Positioning the axis label

Labels can be positioned by using offset and position properties in the labelStyle. The offset defines the distance between the labels and ticks. By default, the offset value is **0**. The possible values of the position are **Inside**, **Outside**, **Cross**, and **Auto**. By default, the labels will be placed inside the axis.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge >
        <e-axes>
          <e-axis :labelStyle='labelStyle2'>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
      labelStyle2: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        position: "Cross"
      }
    }
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs10" %}

Customizing the display of the last label

If the last label is not in the visible range, it will be hidden by default. The last label can be made visible by setting the showLastLabel property as **true** in the e-axis.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge >
        <e-axes>
          <e-axis :showLastLabel=true maximum=115>
            <e-pointers>
              <e-pointer value=20 height=15 width=15 color='#757575'
offset=30></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
    }
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs11" %}

Copyright © 2001 -2024 Syncfusion Inc.

Label Format

Axis labels in the Linear Gauge control can be formatted using the format property in labelStyle. It is used to render the axis labels in a certain format or to add a user-defined unit in the label. It works with the help of placeholder like **{value}°C**, where **value** represents the axis value. For example, 20°C.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge >
        <e-axes>
          <e-axis :labelStyle='labelStyle'>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
      labelStyle: {
        format: "{value}°C"
      }
    }
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs12" %}

Displaying numeric format in labels

The numeric formats such as currency, percentage, and so on can be displayed in the labels of the Linear Gauge using the format property in the Linear Gauge component. The following table describes the result of applying some commonly used label formats on numeric values.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The number is rounded to 1 decimal place.
1000	n2	1000.00	The number is rounded to 2 decimal place.
1000	n3	1000.000	The number is rounded to 3 decimal place.

Copyright © 2001 -2024 Syncfusion Inc.

0.01	p1	1.0%	The number is converted to percentage with 1 decimal place.
0.01	p2	1.00%	The number is converted to percentage with 2 decimal place.
0.01	p3	1.000%	The number is converted to percentage with 3 decimal place.
1000	c1	$1,000.0	The currency symbol is appended to number and number is rounded to 1 decimal place.
1000	c2	$1,000.00	The currency symbol is appended to number and number is rounded to 2 decimal place.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge :format='format'>
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
            format: 'c'
        }
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs13" %}

Orientation

By default, the Linear Gauge is rendered vertically. To change its orientation, the orientation property must be set to **Horizontal**.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
```

```
      <ejs-lineargauge :orientation='orientation' :axes= 'axes' >
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
      orientation: "Horizontal",
      axes: [{
        minimum: 20,
        maximum: 140,
      }]
    }
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs14" %}

Inverted Axis

The axis of the Linear Gauge component can be inversed by setting the isInversed property to **true** in the e-axis.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge >
        <e-axes>
          <e-axis isInversed=true>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
    }
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs15" %}

Opposed Axis

To place an axis opposite from its original position, opposedPosition property in the e-axis must be set as **true**.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge >
        <e-axes>
          <e-axis opposedPosition= true></e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default { };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs16" %}

Multiple Axes

Multiple axes can be added to the Linear Gauge by adding multiple e-axis in the e-axes and customization can be done with the e-axis. Each axis can be customized separately as shown in the following example.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge >
        <e-axes>
          <e-axis :labelStyle='labelStyle'></e-axis>
          <e-axis :opposedPosition='opposedPosition'
:labelStyle='labelStyle1'></e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
      labelStyle: {
        format: '{value}°C'
      },
      opposedPosition: true,
      labelStyle1: {
        format: '{value}°F'
      }
    }
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs17" %}

Ranges in Vue Linear gauge component

Range is the set of values in the axis. The range can be defined using the start and end properties in the e-range. Any number of ranges can be added to the Linear Gauge using the e-ranges.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge>
                <e-axes>
                    <e-axis>
                        <e-ranges>
                           <e-range start=50 end=80></e-range>
                        </e-ranges>
                    </e-axis>
                </e-axes>
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
        }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs64" %}

Customizing the range

Ranges can be customized using the following properties in e-range.

- startWidth - Customize the range thickness at the start axis value.
- endWidth - Customize the range thickness at the end axis value.
- color - Customize the range color.
- position - To place the range. By default, the range is placed outside of the axis. To change the position, this property can be set as "**Inside**", "**Outside**", "**Cross**", or "**Auto**".
- Offset - To place the range with specified distance from the axis.
- border - Customize color and width of range border.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge>
                <e-axes>
                    <e-axis>
                        <e-ranges>
                           <e-range start=50 end=80 startWidth=10
endWidth=20 color="red"></e-range>
                        </e-ranges>
                    </e-axis>
                </e-axes>
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
        }
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
```

```
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs65" %}

Setting the range color for the labels

To set the color of the labels like the range color, set the useRangeColor property as **true** in the labelStyle.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge>
                <e-axes>
                    <e-axis :labelStyle='labelStyle'>
                        <e-ranges>
                            <e-range start=50 end=80 color="red"></e-range>
                        </e-ranges>
                    </e-axis>
                </e-axes>
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
            labelStyle: {
                useRangeColor: true
            }
        }
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs66" %}

Multiple ranges

Multiple ranges can be added to the Linear Gauge by adding collections e-range in the e-ranges and customization of ranges can be done with the e-range.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <ejs-lineargauge>
                <e-axes>
                    <e-axis >
                        <e-ranges>
                            <e-range start= 0 end= 30 startWidth= 10
endWidth= 10  color= '#41f47f'></e-range>
                            <e-range start= 30 end= 50 startWidth= 10
endWidth= 10  color= '#f49441'></e-range>
                            <e-range start= 50 end= 100 startWidth= 10
endWidth= 10  color= '#cd41f4'> </e-range>
                        </e-ranges>
                    </e-axis>
                </e-axes>
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default { };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs67" %}

Gradient Color

Gradient support allows the addition of multiple colors in the range. The following gradient types are supported in the Linear Gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear-gradient, colors will be applied in a linear progression. The start value of the linear gradient can be set using the startValue property. The end value of the linear gradient will be set using the endValue property. The color stop values such as color, opacity, and offset to be defined in colorStop.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge :orientation= 'orientation' :container=
'container'>
                <e-axes>
                    <e-axis :ranges='ranges' :pointers= 'pointers' :line=
'line' :majorTicks= 'majorTicks' :minorTicks= 'minorTicks' :labelStyle=
'labelStyle'></e-axis>
                </e-axes>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, Gradient } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
            orientation: 'Horizontal',
            container: {
                width: 30, offset: 30
            },
            line: { width: 0 },
            majorTicks: { interval: 25, height: 0 },
            minorTicks: { height: 0 },
            labelStyle: {
                font: { color: '#424242',
                }, offset: 55
            },
            pointers: [{
                value: 80, height: 25,
                width: 35, placement: 'Near',
                offset: -44, markerType: 'Triangle',
                color: '#f54ea2'
            }],
            ranges: [{
                start: 0, end: 80,
                startWidth: 30, endWidth: 30,
                offset: 30,
                linearGradient: {
                    startValue: '0%',
                    endValue: '100%',
                    colorStop: [
                        { color: '#fef3f9', offset: '0%', opacity: 1 },
                        { color: '#f54ea2', offset: '100%', opacity: 1 }]
                }
            }]
        }
    },
    provide: {
        lineargauge: [Gradient]
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs68" %}

Copyright © 2001 -2024 Syncfusion Inc.

Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the innerPosition property. The outer circle position of the radial gradient can be set using the outerPosition property. The color stop values such as color, opacity, and offset to be defined in colorStop.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge :orientation= 'orientation' :container=
'container'>
                <e-axes>
                    <e-axis :ranges='ranges' :pointers= 'pointers' :line=
'line' :majorTicks= 'majorTicks' :minorTicks= 'minorTicks' :labelStyle=
'labelStyle'></e-axis>
                </e-axes>
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, Gradient } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
            orientation: 'Horizontal',
            container: {
                width: 30, offset: 30
            },
            line: { width: 0 },
            majorTicks: { interval: 25, height: 0 },
            minorTicks: { height: 0 },
            labelStyle: {
                font: { color: '#424242',
                }, offset: 55
            },
            pointers: [{
                value: 80, height: 25,
                width: 35, placement: 'Near',
                offset: -44, markerType: 'Triangle',
                color: '#f54ea2'
            }],
            ranges: [{
                start: 0, end: 80,
                startWidth: 30, endWidth: 30,
                offset: 30,
                radialGradient: {
                    radius: '65%',
                    outerPosition: { x: '50%', y: '70%' },
                    innerPosition: { x: '60%', y: '60%' },
                    colorStop: [
```

```
                        { color: '#fff5f5', offset: '5%', opacity: 0.9 },
                        { color: '#f54ea2', offset: '100%', opacity: 0.9 }]
                }
            }]
        }
    },
    provide: {
        lineargauge: [Gradient]
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs69" %}

If we set both gradients for the range, only the linear gradient gets rendered. If we set the startValue and endValue of the linearGradient as empty strings, then the radial gradient gets rendered in the pointer of the Linear Gauge.

Pointers in Vue Linear gauge component

Pointers are used to indicate values on an axis. The value of the pointer can be modified using the value property in e-pointer.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=80></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default { };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs55" %}

Types of pointer

The Linear Gauge supports the following types of pointers:

- Bar
- Marker

The type of pointer can be modified by using the type property in e-pointer.

Marker pointer

A marker pointer is a shape that can be used to mark the pointer value in the Linear Gauge.

Types of marker shapes

By default, the marker shape for the pointer is **InvertedTriangle**. To change the shape of the pointer, use the markerType property in e-pointer. The following marker types are available in Linear Gauge.

- Circle
- Rectangle
- Triangle
- InvertedTriangle
- Diamond
- Image
- Text

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=80 type= 'Marker'  markerType= 'Circle'></e-
pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default { };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs56" %}

Image can be rendered instead of rendering a shape as a pointer. It can be achieved by setting the markerType property to **Image** and setting the source URL of image to imageUrl property in e-pointer.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
           <ejs-lineargauge style='display:block' align='center' id='gauge'
orientation="Horizontal">
            <e-axes>
                <e-axis  :majorTicks='majorTicks' :minorTicks='minorTicks'
:labelStyle='labelStyle'>
                    <e-pointers>
                    <e-pointer :value='value' markerType="Image"
imageUrl="https://ej2.syncfusion.com/vue/demos/src/linear-gauge/images/step-
count.png" width="40" height="40" offset="-27"></e-pointer>
                    </e-pointers>
                </e-axis>
            </e-axes>
        </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from '@syncfusion/ej2-vue-lineargauge';
Vue.use(LinearGaugePlugin);
export default {
    data() {
        return {
            minorTicks: {
                position:'Outside'
            },
            majorTicks:{
              interval:20,
              position:'Outside'
            },
            labelStyle: {
                font: {
                    fontFamily: 'inherit'
                },
                position:'Outside'
            },
            value:60
        }
    }
}
</script>
<style>
.wrapper {
    max-width: 400px;
```

```
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs77" %}

Text can be added instead of rendering a shape as a pointer. It can be achieved by setting the markerType property to **Text**, and the text content can be set using the text property in e-pointer.

The following properties in the textStyle property can be used to set the text style for the text pointer.

- fontFamily - It is used to set the font family for the text pointer.
- fontStyle - It is used to set the font style for the text pointer.
- fontWeight - It is used to set the font weight for the text pointer.
- size - It is used to set the font size for the text pointer.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-lineargauge
        minimum="0"
        maximum="100"
        style="display: block"
        align="center"
        id="defaultContainer5"
        :orientation="orientation"
      >
        <e-axes>
          <e-axis
            :majorTicks="majorTicks5"
            :minorTicks="minorTicks5"
            :line="line"
            :ranges="range"
            :labelStyle="labelStyle"
          >
            <e-pointers>
              <e-pointer
                :value="pointerValueOne"
                markerType="Text"
                text="Low"
                color="black"
                :textStyle="textStyle"
                offset="-30"
              ></e-pointer>
              <e-pointer
                :value="pointerValueTwo"
                markerType="Text"
                text="Moderate"
                color="black"
                :textStyle="textStyle"
                offset="-30"
              ></e-pointer>
              <e-pointer
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                :value="pointerValueThree"
                markerType="Text"
                text="High"
                color="black"
                :textStyle="textStyle"
                offset="-30"
              ></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from '@syncfusion/ej2-vue-lineargauge';
Vue.use(LinearGaugePlugin);
export default {
    data() {
        return {
            pointerValueOne: 13,
      pointerValueTwo: 48,
      pointerValueThree: 83,
      orientation: 'Horizontal',
      textStyle: {
        size: '18px',
        fontWeight: 'bold',
      },
      majorTicks5: {
        interval: 20,
        height: 7,
        width: 1,
        position: 'Outside',
      },
      minorTicks5: {
        interval: 10,
        height: 3,
        position: 'Outside',
      },
      labelStyle: { font: { fontFamily: 'inherit' }, position: 'Outside' },
      line: { width: 0 },
      range: [
        {
          start: 0,
          end: 30,
          color: '#FB7D55',
          startWidth: 50,
          endWidth: 50,
        },
        {
          start: 30,
          end: 65,
          color: '#ECC85B',
          startWidth: 50,
          endWidth: 50,
        },
```

```
        {
          start: 65,
          end: 100,
          color: '#6FC78A',
          startWidth: 50,
          endWidth: 50,
        },
      ],
        }
    }
}
</script>
<style>
.wrapper {
    max-width: 400px;
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs78" %}

Marker Pointer Customization

The marker pointer can be customized using the following properties.

- height - To set the height of the pointer.
- position - The position of the pointer can be changed by setting the value as **Inside**, **Outside**, **Cross**, or **Auto**.
- width - To set the width of the pointer.
- color - To set the color of the pointer.
- placement - To place the pointer in the specified position. By default, the pointer is placed **Far** from the axis. To change the placement, set the placement property as **Near**, **Center**, or **None**.
- offset - To place the pointer with specified distance from the axis.
- opacity - To set the opacity of the pointer.
- animationDuration - To specify the duration of the animation in pointer.
- border - To set the color and width for the border of the pointer.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=60 type= 'Marker' markerType= 'Circle'
:height= 15 :width=15 position="Outside" color="#cd41f4"></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default { };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs57" %}

Bar Pointer

The bar pointer is used to track the axis value. The bar pointer starts from the beginning of the gauge and ends at the pointer value. To enable bar pointer, set the type property in e-pointer as **Bar**.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=60 type= 'Bar'></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default { };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs58" %}

Bar pointer customization

The bar pointer can be customized using the following properties.

- width - To set the thickness of the bar pointer.

Copyright © 2001 -2024 Syncfusion Inc.

- color - To set the color of the bar pointer.
- offset - To place the bar pointer with the specified distance from it's default position.
- opacity - To set the opacity of the bar pointer.
- roundedCornerRadius - To set the corner radius for the bar pointer.
- border - To set the color and width for the border of the pointer.
- animationDuration - To set the duration of the animation in bar pointer.

The placement property is not applicable for the bar pointer.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=80 type= 'Bar' color= '#f44141'></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
   </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default { };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs59" %}

Multiple pointers

Multiple pointers can be added to the Linear Gauge by adding multiple e-pointer in the e-pointers and customization for the pointers can be done with e-pointer.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge >
        <e-axes>
          <e-axis >
            <e-pointers>
              <e-pointer value=80></e-pointer>
              <e-pointer value=30 markerType= 'Diamond'></e-pointer>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
              <e-pointer value=50 markerType= 'Circle'></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
   </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default { };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs60" %}

Pointer animation

Pointer is animated on loading the gauge. This can be handled using the animationDuration property. The duration of the animation can be specified in milliseconds.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=80 animationDuration= 1000></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
   </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default { };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs61" %}

Gradient Color

Gradient support allows the addition of multiple colors in the pointers of the Linear Gauge. The following gradient types are supported in the Linear Gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear gradient, colors will be applied in a linear progression. The start value of the linear gradient can be set using the startValue property. The end value of the linear gradient will be set using the endValue property. The color stop values such as color, opacity, and offset are set using colorStop property.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :orientation= 'orientation' :container= 'container'>
        <e-axes>
          <e-axis :ranges='ranges' :pointers= 'pointers' :line= 'line'
:majorTicks= 'majorTicks' :minorTicks= 'minorTicks' :labelStyle=
'labelStyle'></e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, Gradient } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
      orientation: 'Horizontal',
      container: {
        width: 30, offset: 30
      },
      line: { width: 0 },
      majorTicks: { interval: 25, height: 0 },
      minorTicks: { height: 0 },
      labelStyle: {
        font: { color: '#424242'},
        offset: 55
      },
      pointers: [{
        value: 80, height: 25,
        width: 35, placement: 'Near',
        offset: -44, markerType: 'Triangle',
        linearGradient: {
          startValue: '0%',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          endValue: '100%',
          colorStop: [
            { color: '#fef3f9', offset: '0%', opacity: 1 },
            { color: '#f54ea2', offset: '100%', opacity: 1 }]
          }
      }],
      ranges: [{
        start: 0, end: 80,
        startWidth: 30, endWidth: 30,
        color: '#f54ea2', offset: 30,
      }]
    }
  },
  provide: {
    lineargauge: [Gradient]
  }
};
</script>
<style>
#content-wrapper {
  padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs62" %}

Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the innerPosition property. The outer circle position of the radial gradient can be set using the outerPosition property. The color stop values such as color, opacity, and offset are set using colorStop property.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :orientation= 'orientation' :container= 'container'>
      <e-axes>
        <e-axis :ranges='ranges' :pointers= 'pointers' :line= 'line'
:majorTicks= 'majorTicks' :minorTicks= 'minorTicks' :labelStyle=
'labelStyle'></e-axis>
      </e-axes>
    </ejs-lineargauge>
   </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, Gradient } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return {
      orientation: 'Horizontal',
```

```
      container: {
        width: 30, offset: 30
      },
      line: { width: 0 },
      majorTicks: { interval: 25, height: 0 },
      minorTicks: { height: 0 },
      labelStyle: {
        font: { color: '#424242'},
        offset: 55
      },
      pointers: [{
        value: 80, height: 25,
        width: 35, placement: 'Near',
        offset: -44, markerType: 'Triangle',
        radialGradient: {
          radius: '60%',
          outerPosition: { x: '50%', y: '50%' },
          innerPosition: { x: '50%', y: '50%' },
          colorStop: [
            { color: '#fff5f5', offset: '0%', opacity: 0.9 },
            { color: '#f54ea2', offset: '100%', opacity: 0.8 }]
          }
      }],
      ranges: [{
        start: 0, end: 80,
        startWidth: 30, endWidth: 30,
        color: '#f54ea2', offset: 30,
      }]
    }
  },
  provide: {
    lineargauge: [Gradient]
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs63" %}

If we set both gradients, only the linear gradient gets rendered. If we set the startValue and endValue of the linearGradient as empty strings, then the radial gradient gets rendered in the pointer of the Linear Gauge.

Annotations in Vue Linear gauge component

<!-- markdownlint-disable MD013 -->

Annotations are used to mark the specific area of interest in the Linear Gauge with text, HTML elements, or images. Any number of annotations can be added to the Linear Gauge component.

Copyright © 2001 -2024 Syncfusion Inc.

Adding annotation

To render the custom HTML elements in the Linear Gauge component, use the content property in the e-annotation. The annotation can be rendered either by specifying the id of the element or specifying the code to create a new element that needs to be displayed in the gauge area.

<!-- markdownlint-disable MD036 -->

`ts

<template>

<ejs-lineargauge ref='lineargauge' style='display:block' align='center'>

<e-annotations>

<e-annotation :content='contentTemplate' :zIndex='zindex' :axisValue=50>

</e-annotation>

</e-annotations>

</ejs-lineargauge>

</template>

<style>

control-container {

padding: 0px !important;

}

</style>

<script>

import Vue from "vue";

import { LinearGaugePlugin, Annotations } from "@syncfusion/ej2-vue-lineargauge";

Vue.use(LinearGaugePlugin);

export default {

data:function() {

return {

zindex: 1,

contentTemplate: ' <h1>Gauge</h1> ',

}

},

Copyright © 2001 -2024 Syncfusion Inc.

provide: {

lineargauge: [Annotations]

},

};

</script>

`

Customization

The following properties are used to customize the annotation.

- zIndex - Bring the annotation to the front or back, when annotation overlaps with another element.
- axisValue - To place the annotation in the specified axis value with respect to the provided axis index.
- axisIndex - To place the annotation in the specified axis with respect to the provided axis value.
- horizontalAlignment - To place the annotation horizontally.
- verticalAlignment - To place the annotation vertically.
- x, y - To place the annotation in the specified location.

Changing the z-index

To change the stack order of an annotation element, the zIndex property of the e-annotation can be used.

APP.VUE

```
<template>
  <div class="control-section">
    <div align='center'>
      <ejs-lineargauge style='display:block' align='center'>
        <e-annotations>
          <e-annotation :content='contentTemplate' :zIndex='zindex'
horizontalAlignment='Center' verticalAlignment='Center'>
          </e-annotation>
        </e-annotations>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { LinearGaugePlugin, Annotations } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data:function(){
    return{
      zindex: 1,
      contentTemplate: '<div id="first"><h1>Gauge</h1></div>',
    }
  },
  provide: {
```

```
    lineargauge: [Annotations]
  },
};
</script>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs1" %}

Positioning an annotation

The annotation can be placed anywhere in the Linear Gauge by setting the pixel value to the x and y properties in the e-annotation.

APP.VUE

```
<template>
  <div class="control-section">
    <div align='center'>
      <ejs-lineargauge style='display:block' align='center'>
        <e-annotations>
          <e-annotation :content='contentTemplate' :zIndex='zindex' x=100
y=100>
          </e-annotation>
        </e-annotations>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { LinearGaugePlugin, Annotations } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data:function() {
    return{
      zindex: 1,
      contentTemplate: '<div id="first"><h1>Gauge</h1></div>',
    }
  },
  provide: {
    lineargauge: [Annotations]
  },
};
</script>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs2" %}

Alignment of annotation

The annotation can be aligned horizontally and vertically by using the horizontalAlignment and verticalAlignment properties respectively. The possible values can be **Center**, **Far**, **Near**, and **None**. The horizontalAlignment and verticalAlignment properties are not applicable when the x and y properties are set in the e-annotation.

APP.VUE

```
<template>
  <div class="control-section">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    <div align='center'>
      <ejs-lineargauge style='display:block' align='center'>
        <e-annotations>
          <e-annotation :content='contentTemplate' :zIndex='zindex'
horizontalAlignment='Center' verticalAlignment='Center'>
          </e-annotation>
        </e-annotations>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { LinearGaugePlugin, Annotations } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data:function() {
    return {
      zindex: 1,
      contentTemplate: '<div id="first"><h1>Gauge</h1></div>',
    }
  },
  provide: {
    lineargauge: [Annotations]
  },
};
</script>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs3" %}

Multiple annotations

Multiple annotations can be added to the Linear Gauge component by adding the multiple e-annotation in the e-annotations and customization for the annotation can be done with the e-annotation.

APP.VUE

```
<template>
  <div class="control-section">
    <div align='center'>
      <ejs-lineargauge style='display:block' align='center'>
        <e-annotations>
          <e-annotation :content='contentTemplate' :zIndex='zindex'
horizontalAlignment='Near' verticalAlignment='Center' :x='x1' :y='y1'>
          </e-annotation>
          <e-annotation :content='contentTemplate1' :zIndex='zindex'
verticalAlignment='Center' :x='x2' :y='y2'>
          </e-annotation>
        </e-annotations>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { LinearGaugePlugin, Annotations } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data:function() {
    return {
      zindex: 1,
      contentTemplate: '<div><h1 style="color:red;">Speed</h1></div>',
      contentTemplate1: '<div><h1 style="color:blue;">Meter</h1></div>',
      x1:290,
      y1:250,
      x2:10,
      y2:-190,
    }
  },
  provide: {
    lineargauge: [Annotations]
  },
};
</script>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs4" %}

Animation in Vue Linear Gauge component

All of the elements in the Linear Gauge, such as the axis lines, ticks, labels, ranges, pointers, and annotations, can be animated sequentially by using the animationDuration property. The animation for the Linear Gauge is enabled when the animationDuration property is set to an appropriate value in milliseconds, providing a smooth rendering effect for the component. If the animationDuration property is set to **0**, which is the default value, the animation effect is disabled. If the animation is enabled, the component will behave in the following order.

1. The axis line, ticks, labels, and ranges will all be animated at the same time.
2. If available, pointers will be animated in the same way as pointer animation.
3. If available, annotations will be animated.

The animation of the Linear Gauge is demonstrated in the following example.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-lineargauge animationDuration='3000'
style='display:block' align='center' id='defaultContainer'
:orientation='orientation'>
                <e-annotations>
                    <e-annotation :axisIndex='axisIndex'
:axisValue='axisValue' :x='positionX' zIndex='1' :y='positionY'
:zIndex='zIndex' :content="content">
                    </e-annotation>
                </e-annotations>
                <e-axes>
                    <e-axis :majorTicks='majorTicks'
:minorTicks='minorTicks' :labelStyle='labelStyle' >
```

```
                        <e-pointers>
                            <e-pointer :value='value' :height='height'
:placement='placement' :width='width' :offset='offset'
:markerType='markerType' ></e-pointer>
                        </e-pointers>
                        <e-ranges>
                        <e-range start=0 end="50" startWidth="10"
endWidth="10" color="#F45656" offset="35"></e-range>
                        </e-ranges>
                    </e-axis>
                </e-axes>
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, Annotations } from '@syncfusion/ej2-vue-
lineargauge';
Vue.use(LinearGaugePlugin);
export default {
    data() {
        return {
          content:'<div id="pointer" style="width: 70px;margin-left:-
25%;margin-top: 42%;font-size: 16px;">10 MPH</div>',
        orientation:"Horizontal",
        value:10,
        height:15,
        width:15,
        offset:-40,
        markerType:'Triangle',
        placement:'Near',
        majorTicks: {
            interval: 10,
            height:20,
            color:'#9E9E9E'
        },
        minorTicks:{
            interval: 2,
            height:10,
            color:'#9E9E9E'
        },
        labelStyle:{
            offset: 48,
            font: {
                fontFamily: 'inherit'
            }
        },
        axisIndex: 0,
        axisValue: 10,
        positionX: 10,
        positionY: -70,
        zIndex:'1'
        }
    },
     provide: {
        lineargauge: [Annotations]
```

```
    },
}
</script>
<style>
.wrapper {
    max-width: 400px;
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs76" %}

Only the pointer of the Linear Gauge can be animated individually, not the axis lines, ticks, labels, ranges, and annotations. You can refer this link to enable only pointer animation.

User interaction in Vue Linear gauge component

Tooltip

<!-- markdownlint-disable MD036 -->

Linear gauge displays the details about a pointer value through tooltip, when the mouse hovers over the pointer. To enable the tooltip, set the enable property to **true** and inject the **GaugeTooltip** module in **provide** section.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge   :tooltip='tooltip' >
        <e-axes>
          <e-axis >
            <e-pointers>
              <e-pointer value=80 ></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, GaugeTooltip } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return{
      tooltip: {
        enable: true
      }
    }
  },
  provide: {
    lineargauge: [ GaugeTooltip]
  },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs70" %}

Tooltip Format

Tooltip in the Linear Gauge control can be formatted using the format property in tooltip object. It is used to render the tooltip in certain format or to add a user-defined unit in the tooltip. By default, the tooltip shows the pointer value only. In addition to that, more information can be added in the tooltip. For example, the format **{value}km** shows pointer value with kilometer unit in the tooltip.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :tooltip='tooltip'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=80 ></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, GaugeTooltip } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return{
      tooltip: {
        enable: true,
        format: '{value}km'
      }
    }
  },
  provide: {
    lineargauge: [ GaugeTooltip]
  },
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
```

```
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs71" %}

Tooltip Template

The HTML element can be rendered in the tooltip of the Linear Gauge using the template property in the tooltip. The **${value}** can be used as placeholders in the HTML element to display the pointer values of the corresponding axis.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :tooltip='tooltip'>
        <e-axes>
          <e-axis >
            <e-pointers>
              <e-pointer value=80></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, GaugeTooltip } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return{
      tooltip: {
        enable: true,
        template: '<div>Pointer: 80 </div>'
      }
    }
  },
  provide: {
    lineargauge: [GaugeTooltip]
  },
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs72" %}

Customize the appearance of the tooltip

The tooltip can be customized using the following properties in tooltip.

- fill - To fill the color for tooltip.
- enableAnimation - To enable or disable the tooltip animation.
- border - To set the border color and width of the tooltip.
- textStyle - To customize the style of the text in tooltip.
- showAtMousePosition - To show the tooltip at the mouse position.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :tooltip='tooltip'>
        <e-axes>
          <e-axis >
            <e-pointers>
              <e-pointer value=80></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, GaugeTooltip } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return{
      tooltip: {
        enable: true,
        fill: '#e5bcbc',
        border: {
          color: '#d80000',
          width: 2
        }
      }
    }
  },
  provide: {
    lineargauge: [ GaugeTooltip]
  },
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs73" %}

Copyright © 2001 -2024 Syncfusion Inc.

Positioning the tooltip

The tooltip is positioned at the **End** of the pointer. To change the position of the tooltip at the start, or center of the pointer, set the position property to **Start** or **Center**.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :tooltip='tooltip'>
        <e-axes>
          <e-axis >
            <e-pointers>
              <e-pointer value=80 type="Bar" color="blue"></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, GaugeTooltip } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data: function () {
    return{
      tooltip: {
        enable: true,
        position: "Center"
      }
    }
  },
  provide: {
    lineargauge: [ GaugeTooltip]
  },
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs74" %}

Pointer Drag

To drag either marker or bar pointer to the desired axis value, set the enableDrag property as **true** in the e-pointer.

APP.VUE

```
<template>
  <div class="content-wrapper">
```

```
    <div align='center'>
      <ejs-lineargauge >
        <e-axes>
          <e-axis >
            <e-pointers>
              <e-pointer value=80  enableDrag= 'true' ></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default { };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs75" %}

Linear gauge print and export in Vue Linear gauge component

Print

The rendered Linear Gauge can be printed directly from the browser by calling the print method. To use the print functionality, set the allowPrint property as **true** and inject the **Print** module in **provide** section.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <button v-on:click="print">Print</button>
      <ejs-lineargauge id="container" ref='gauge' :allowPrint='allowPrint'>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, Print } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data () {
    return {
      allowPrint: true
    }
  },
  methods: {
```

```
    print: function (event) {
      this.$refs.gauge.ej2Instances.print();
    }
  },
  provide: {
    lineargauge: [Print]
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs48" %}

Export

Image Export

To use the image export functionality, set the allowImageExport property as **true** and inject the **ImageExport** module in **provide** section. The rendered Linear Gauge can be exported as an image using the export method. This method requires two parameters: export type and file name. The Linear Gauge can be exported as an image with the following formats.

- JPEG
- PNG
- SVG

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <button v-on:click="clickExport">Image Export</button>
      <ejs-lineargauge id="container" ref='gauge'
:allowImageExport='allowImageExport'>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, ImageExport } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data () {
    return {
      allowImageExport: true
    }
  },
  methods: {
    clickExport: function (event) {
      this.$refs.gauge.ej2Instances.export("PNG", "Gauge");
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
  },
  provide: {
    lineargauge: [ImageExport]
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs49" %}

PDF Export

To use the PDF export functionality, set the allowPdfExport property as **true** and inject the **PdfExport** module in **provide** section. The rendered Linear Gauge can be exported as PDF using the export method. The export method requires three parameters: file type, file name, and orientation of the PDF document. The orientation of the PDF document can be set as **Portrait** or **Landscape**.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <button v-on:click="clickExport">PDF Export</button>
      <ejs-lineargauge id="container" ref='gauge'
:allowPdfExport='allowPdfExport'>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, PdfExport } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data () {
    return {
      allowPdfExport: true
    }
  },
  methods: {
    clickExport: function (event) {
      this.$refs.gauge.ej2Instances.export("PNG", "Gauge");
    }
  },
  provide: {
    lineargauge: [PdfExport]
  }
};
</script>
<style>
#content-wrapper {
```

```
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs50" %}

Exporting Linear Gauge as base64 string of the file

The Linear Gauge can be exported as base64 string for the JPEG, PNG and PDF formats. The rendered Linear Gauge can be exported as base64 string of the exported image or PDF document used in the export method. The arguments that are required for this method is export type, file name, orientation of the exported PDF document and **allowDownload** boolean value that is set as **false** to return base64 string. The value for the orientation of the exported PDF document is set as **null** for image export and **Portrait** or **Landscape** for the PDF document.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <button v-on:click="clickExport">Export</button>
      <ejs-lineargauge id="container" ref='gauge'
:allowImageExport='allowImageExport'>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, ImageExport } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data () {
    return {
      allowImageExport: true
    }
  },
  methods: {
    clickExport: function (event) {
      this.$refs.gauge.ej2Instances.export('JPEG', 'Gauge', null,
false).then((data) => {
            let base64 = data;
            document.writeln(base64);
        });
    }
  },
  provide: {
    lineargauge: [ImageExport]
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs51" %}

The exporting of the Linear Gauge as base64 string is not applicable for the SVG format.

Linear gauge appearance in Vue Linear gauge component

Customizing the Linear Gauge area

The following properties are available in the ejs-lineargauge to customize the Linear Gauge area.

- background - Applies the background color for the Linear gauge.
- border - To customize the color and width of the border in Linear Gauge.
- margin - To customize the margins of the Linear Gauge.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge :margin='margin' background= 'skyblue':border
='border'>
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
            border: { color: "#FF0000", width: 2 },
            margin: {
                left: 20,
                top: 20,
                right: 20,
                bottom: 20
             }
        }
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs40" %}

Setting up the Linear Gauge title

The title for the Linear Gauge can be set using title property in Linear Gauge. Its appearance can be customized using the titleStyle with the below properties.

Copyright © 2001 -2024 Syncfusion Inc.

- color - Specifies the text color of the title.
- fontFamily - Specifies the font family of the title.
- fontStyle - Specifies the font style of the title.
- fontWeight - Specifies the font weight of the title.
- opacity - Specifies the opacity of the title.
- size - Specifies the font size of the title.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge :title ='title' :titleStyle ='titleStyle'>
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
            title: 'linear gauge',
            titleStyle: {
                fontFamily: "Arial",
                fontStyle: 'italic',
                fontWeight: 'regular',
                color: "#E27F2D",
                size: '23px'
            }
        }
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs41" %}

Customizing the Linear Gauge container

The area used to render the ranges and pointers at the center position of the gauge is called container. The following types of container to be applicable for Linear Gauge.

- Normal
- Rounded Rectangle
- Thermometer

Copyright © 2001 -2024 Syncfusion Inc.

The type of the container can be modified by using the type property in container. The container can be customized by using the following properties in container.

- offset - To place the container with the specified distance from the axis of the Linear Gauge.
- width - To set the thickness of the container.
- height - To set the length of the container.
- backgroundColor - To set the background color of the container.
- border - To set the color and width for the border of the container.

Normal

The **Normal** type will render the container as a rectangle and this is the default container type.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge :container='container'>
                <e-axes>
                    <e-axis >
                        <e-pointers>
                            <e-pointer value= 50 width= 15 type= 'Bar'></e-
pointer>
                        </e-pointers>
                    </e-axis>
                </e-axes>
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
            container: {
               width: 30
            }
        }
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs42" %}

Copyright © 2001 -2024 Syncfusion Inc.

Rounded Rectangle

The **RoundedRectangle** type will render the container as a rectangle with rounded corner radius. The rounded corner radius of the container can be customized using the roundedCornerRadius property in container.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge :container='container'>
                <e-axes>
                    <e-axis>
                        <e-pointers>
                            <e-pointer value= 50  type= 'Bar'></e-pointer>
                        </e-pointers>
                    </e-axis>
                </e-axes>
            </ejs-lineargauge>
        </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
            container: {
               width: 30,
               type: 'RoundedRectangle'
            }
        }
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs43" %}

Thermometer

The **Thermometer** type will render the container similar to the appearance of thermometer.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge :container='container'>
                <e-axes>
                    <e-axis >
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                        <e-pointers>
                            <e-pointer value= 50  type= 'Bar'></e-pointer>
                        </e-pointers>
                    </e-axis>
                </e-axes>
            </ejs-lineargauge>
        </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
            container: {
               width: 30,
               type: 'Thermometer'
            }
        }
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs44" %}

Fitting Linear Gauge to the control

The Linear Gauge component is rendered with margin by default. To remove the margin around the Linear Gauge, the allowMargin property in the ejs-lineargauge is set as **false**.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge orientation='Horizontal' :allowMargin=false
:margin='margin' background='skyblue':border ='border'>
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
            border: { color: "#FF0000", width: 2 },
            margin: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                left: 0,
                top: 0,
                right: 0,
                bottom: 0
            }
        }
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs45" %}

To use this feature, set the allowMargin property to **false**, the width property to **100%** and the properties of margin to **0**.

Accessibility in Vue Linear Gauge component

Linear Gauge has built-in accessibility features like screen reading and WAI-ARIA attributes.

WAI-ARIA attributes

The Linear Gauge component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Linear Gauge component:

| Attributes | Purpose |

| --- | --- |

| role=region | It is specified in the pointer where the interactive drag and drop function is supported to update the pointer value. |

| aria-label | Provides an accessible name for the axis labels, text pointer and annotation. |

Screen reading in Linear Gauge

Accessibility in the Linear Gauge component ensures that all users, regardless of ability or disability, can use screen reading. The following Linear Gauge elements will be read aloud using screen reading software, such as Narrator for Windows.

| Elements | Description |

| --- | --- |

| Axis labels | Reads the axis labels of the Linear Gauge.|

| Text pointer | Reads the text content shown as a pointer in Linear Gauge. |

| Annotation | Reads the content specified in the annotation. |

Ensuring accessibility

The Linear Gauge component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

Copyright © 2001 -2024 Syncfusion Inc.

The accessibility compliance of the Linear Gauge component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Linear Gauge component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/linear-gauge.html" %}

See also

- Accessibility in Syncfusion Vue components

Internationalization in Vue Linear gauge component

Globalization is the process of designing and developing a component that works in different cultures. Internationalization is used to globalize the number content in Linear Gauge component using format property in Linear Gauge. It has static text on some features such as

- Axis label
- Tooltip

The static text on above features can be changed to any culture such as Arabic, Deutsch and French. To know more about the globalization in Vue components, refer here.

Numeric Format

The text in axis labels and tooltip can be displayed in the numeric format such as currency, percentage and so on. To know more about the numeric formats in axis labels, refer here. In the below example, the axis label is displayed in the currency format.

APP.VUE

```
<template>
    <div class="content-wrapper">
        <div align='center'>
            <ejs-lineargauge :format='format'>
            </ejs-lineargauge>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
    data: function () {
        return {
            format: 'c'
        }
    }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs39" %}

Events in Vue Linear gauge component

This section describes the Linear Gauge component's event that gets triggered when corresponding operations are performed.

animationComplete

When the pointer animation is completed, the animationComplete event will be triggered. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :animationComplete='animationComplete'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=10></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    animationComplete: function (event) {
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs18" %}

annotationRender

Before the annotation is rendered in the Linear Gauge, the annotationRender event will be triggered. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :annotationRender='annotationRender'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <e-annotations>
            <e-annotation :content='contentTemplate' :zIndex='zindex'
:axisValue='axisValue'>
            </e-annotation>
        </e-annotations>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=10></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, Annotations } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data:function(){
    return{
        zindex: 1,
        contentTemplate : '10',
        axisValue: 0
    }
  },
  methods: {
    annotationRender: function (event) {
    }
  },
  provide: {
    lineargauge: [Annotations]
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs19" %}

axisLabelRender

Before each axis label is rendered in the Linear Gauge, the axisLabelRender event is fired. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :axisLabelRender='axisLabelRender'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    axisLabelRender: function (event) {
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs20" %}

beforePrint

The beforePrint event is fired before the print begins. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <button v-on:click="print">Print</button>
      <ejs-lineargauge id="container" ref='gauge' :allowPrint='allowPrint'
:beforePrint='beforePrint'>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, Print } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data () {
    return {
      allowPrint: true
    }
  },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  methods: {
    beforePrint: function (event) {
    },
    print: function (event) {
      this.$refs.gauge.ej2Instances.print();
    }
  },
  provide: {
    lineargauge: [Print]
  }
};
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs21" %}

dragEnd

The dragEnd event will be fired before the pointer drag is completed. To know more about the argument of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :dragEnd='dragEnd'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer enableDrag=true></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    dragEnd: function (event) {
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
```

```
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs22" %}

dragMove

The dragMove event will be fired when the pointer is dragged. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :dragMove='dragMove'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer enableDrag=true></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    dragMove: function (event) {
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs23" %}

dragStart

When the pointer drag begins, the dragStart event is triggered. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :dragStart='dragStart'>
        <e-axes>
          <e-axis>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <e-pointers>
              <e-pointer enableDrag=true></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    dragStart: function (event) {
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs24" %}

gaugeMouseDown

When mouse is pressed down on the gauge, the gaugeMouseDown event is triggered. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :gaugeMouseDown='gaugeMouseDown'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    gaugeMouseDown: function (event) {
```

```
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs25" %}

gaugeMouseLeave

When mouse pointer moves over the gauge, the gaugemouseleave event is triggered. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :gaugeMouseLeave='gaugeMouseLeave'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    gaugeMouseLeave: function (event) {
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs26" %}

gaugeMouseMove

When mouse pointer leaves the gauge, the gaugeMouseMove event is triggered. To know more about the arguments of this event, refer here.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :gaugeMouseMove='gaugeMouseMove'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    gaugeMouseMove: function (event) {
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs27" %}

gaugeMouseUp

When the mouse pointer is released over the Linear Gauge, the gaugeMouseUp event is triggered. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :gaugeMouseUp='gaugeMouseUp'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    gaugeMouseUp: function (event) {
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs28" %}

load

Before the Linear Gauge is loaded, the load event is fired. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :load='load'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    load: function (event) {
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
```

```
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs29" %}

loaded

After the Linear Gauge has been loaded, the loaded event will be triggered. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :loaded='loaded'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    loaded: function (event) {
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs30" %}

resized

After the window resizing, the resized event is triggered. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :resized='resized'>
        <e-axes>
          <e-axis>
```

```
            <e-pointers>
              <e-pointer></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    resized: function (event) {
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs31" %}

tooltipRender

The tooltipRender event is fired before the tooltip is rendered. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :tooltip='tooltip' :tooltipRender='tooltipRender'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, GaugeTooltip  } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data:function(){
```

```
    return {
      tooltip: {
        enable: true,
      }
    }
  },
  methods: {
    tooltipRender: function (event) {
    }
  },
  provide: {
    lineargauge: [GaugeTooltip]
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs32" %}

valueChange

The valueChange event is triggered when the pointer is dragged from one value to another. To know more about the arguments of this event, refer here.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <ejs-lineargauge :valueChange='valueChange'>
        <e-axes>
          <e-axis>
            <e-pointers enableDrag=true>
              <e-pointer></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    valueChange: function (event) {
    }
  }
 };
</script>
<style>
```

```
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs33" %}

Methods in Vue Linear gauge component

The following methods are available in the Linear Gauge component.

setPointerValue

To change the pointer value dynamically, use the setPointerValue method in the Linear Gauge component. The following are the arguments for this method.

Argument name	Description
axisIndex	Specifies the index of the axis in which the pointer value is to be updated.
pointerIndex	Specifies the index of the pointer to be updated.
pointerValue	Specifies the value of the pointer to be updated.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <button v-on:click="clicked">Click</button>
      <ejs-lineargauge ref='gauge'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=80></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    clicked: function (event) {
      this.$refs.gauge.ej2Instances.setPointerValue(0, 0, 30);
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs52" %}

setAnnotationValue

To change the annotation content dynamically, use the setAnnotationValue method in the Linear Gauge component. The following are the arguments for this method.

Argument name	Description
annotationIndex	Specifies the index number of the annotation to be updated.
content	Specifies the text for the annotation to be updated.
axisValue	Specifies the value of the axis where the annotation is to be placed.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <button v-on:click="clicked">Click</button>
      <ejs-lineargauge ref='gauge'>
        <e-annotations>
            <e-annotation :content='contentTemplate' :zIndex='zindex'
:axisValue='axisValue'>
            </e-annotation>
        </e-annotations>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=10></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin, Annotations } from "@syncfusion/ej2-vue-
lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  data:function(){
    return{
        zindex: 1,
        contentTemplate : '10',
        axisValue: 0
    }
  },
  methods: {
    clicked: function (event) {
      this.$refs.gauge.ej2Instances.setAnnotationValue(0, '50', 50);
```

```
    }
  },
  provide: {
    lineargauge: [Annotations]
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs53" %}

refresh

The refresh method can be used to change the state of the component and render it again.

APP.VUE

```
<template>
  <div class="content-wrapper">
    <div align='center'>
      <button v-on:click="clicked">Click</button>
      <ejs-lineargauge ref='gauge'>
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=10></e-pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-lineargauge>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { LinearGaugePlugin } from "@syncfusion/ej2-vue-lineargauge";
Vue.use(LinearGaugePlugin);
export default {
  methods: {
    clicked: function (event) {
      this.$refs.gauge.ej2Instances.refresh();
    }
  }
 };
</script>
<style>
#content-wrapper {
    padding: 0px !important;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/linear-gauge/getting-started-cs54" %}

Copyright © 2001 -2024 Syncfusion Inc.

ListBox

Getting Started with the Vue List box Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue List box component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

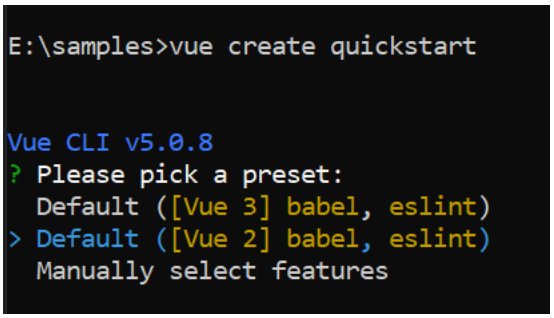
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option `Default ([Vue 2] babel, eslint)` from the menu.



Once the `quickstart` project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue List box component as an example. Install the @syncfusion/ej2-vue-dropdowns package by running the following command:

`bash

npm install @syncfusion/ej2-vue-dropdowns --save

`

or

`bash

yarn add @syncfusion/ej2-vue-dropdowns

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the List box component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue List box component using Composition API or Options API:

1\. First, import and register the List box component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { ListBoxComponent as EjsListbox } from "@syncfusion/ej2-vue-
dropdowns";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { ListBoxComponent } from "@syncfusion/ej2-vue-dropdowns";
export default {
components: {
'ejs-listbox': ListBoxComponent
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
```

2\. In the template section, define the List box component with the dataSource property.

~/SRC/APP.VUE

```
<template>
<div id="app">
<div id='container' style="margin:10px auto 0; width:250px;">
<ejs-listbox :dataSource='data' ></ejs-listbox>
</div>
</div>
</template>
```

Binding data source

After initialization, populate the ListBox with data using the dataSource property. Here, an array of object is passed to the ListBox component.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div id='container' style="margin:10px auto 0; width:250px;">
<ejs-listbox :dataSource='data' ></ejs-listbox>
</div>
</div>
</template>
<script setup>
import { ListBoxComponent as EjsListbox } from "@syncfusion/ej2-vue-
dropdowns";
const data = [
{ text: 'Hennessey Venom', id: 'list-01' },
{ text: 'Bugatti Chiron', id: 'list-02' },
{ text: 'Bugatti Veyron Super Sport', id: 'list-03' },
{ text: 'SSC Ultimate Aero', id: 'list-04' },
{ text: 'Koenigsegg CCR', id: 'list-05' },
{ text: 'McLaren F1', id: 'list-06' },
{ text: 'Aston Martin One- 77', id: 'list-07' },
{ text: 'Jaguar XJ220', id: 'list-08' },
{ text: 'McLaren P1', id: 'list-09' },
{ text: 'Ferrari LaFerrari', id: 'list-10' },
];
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div id='container' style="margin:10px auto 0; width:250px;">
<ejs-listbox :dataSource='data' ></ejs-listbox>
</div>
</div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { ListBoxComponent } from "@syncfusion/ej2-vue-dropdowns";
export default {
components: {
'ejs-listbox': ListBoxComponent
},
data (){
return {
data: [
{ text: 'Hennessey Venom', id: 'list-01' },
{ text: 'Bugatti Chiron', id: 'list-02' },
{ text: 'Bugatti Veyron Super Sport', id: 'list-03' },
{ text: 'SSC Ultimate Aero', id: 'list-04' },
{ text: 'Koenigsegg CCR', id: 'list-05' },
{ text: 'McLaren F1', id: 'list-06' },
{ text: 'Aston Martin One- 77', id: 'list-07' },
{ text: 'Jaguar XJ220', id: 'list-08' },
{ text: 'McLaren P1', id: 'list-09' },
{ text: 'Ferrari LaFerrari', id: 'list-10' },
]
}
}
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data' ></ejs-listbox>
    </div>
  </div>
</template>
<script setup>
import { ListBoxComponent as EjsListbox } from "@syncfusion/ej2-vue-
dropdowns";
const data = [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data' ></ejs-listbox>
    </div>
  </div>
</template>
<script>
import { ListBoxComponent } from "@syncfusion/ej2-vue-dropdowns";
export default {
  components: {
    'ejs-listbox': ListBoxComponent
  },
  data (){
    return {
       data: [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
]
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs9" %}

Getting Started with the Vue ListBox Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue ListBox component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Copyright © 2001 -2024 Syncfusion Inc.

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue ListBox component as an example. To use the Vue ListBox component in the project, the @syncfusion/ej2-vue-dropdowns package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-dropdowns --save

`

or

`bash

yarn add @syncfusion/ej2-vue-dropdowns

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the ListBox component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue ListBox component using Composition API or Options API:

1.First, import and register the ListBox component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { ListBoxComponent as EjsListbox } from "@syncfusion/ej2-vue-
dropdowns";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { ListBoxComponent } from "@syncfusion/ej2-vue-dropdowns";
//Component registration
export default {
name: "App",
components: {
"ejs-listbox": ListBoxComponent
}
}
</script>
```

2.In the template section, define the ListBox component with the dataSource property.

~/SRC/APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<div class="control_wrapper">
<ejs-listbox :dataSource='data'></ejs-listbox>
</div>
</template>
```

3.Declare the values for the dataSource property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const data = [
{ text: 'Hennessey Venom', id: 'list-01' },
{ text: 'Bugatti Chiron', id: 'list-02' },
{ text: 'Bugatti Veyron Super Sport', id: 'list-03' },
{ text: 'SSC Ultimate Aero', id: 'list-04' },
{ text: 'Koenigsegg CCR', id: 'list-05' },
{ text: 'McLaren F1', id: 'list-06' },
{ text: 'Aston Martin One- 77', id: 'list-07' },
{ text: 'Jaguar XJ220', id: 'list-08' },
{ text: 'McLaren P1', id: 'list-09' },
{ text: 'Ferrari LaFerrari', id: 'list-10' }
]
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
data:[
{ text: 'Hennessey Venom', id: 'list-01' },
{ text: 'Bugatti Chiron', id: 'list-02' },
{ text: 'Bugatti Veyron Super Sport', id: 'list-03' },
{ text: 'SSC Ultimate Aero', id: 'list-04' },
{ text: 'Koenigsegg CCR', id: 'list-05' },
{ text: 'McLaren F1', id: 'list-06' },
{ text: 'Aston Martin One- 77', id: 'list-07' },
{ text: 'Jaguar XJ220', id: 'list-08' },
{ text: 'McLaren P1', id: 'list-09' },
{ text: 'Ferrari LaFerrari', id: 'list-10' }
],
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-listbox :dataSource='data'></ejs-listbox>
</div>
</template>
<script setup>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { ListBoxComponent as EjsListbox } from "@syncfusion/ej2-vue-
dropdowns";
const data = [
{ text: 'Hennessey Venom', id: 'list-01' },
{ text: 'Bugatti Chiron', id: 'list-02' },
{ text: 'Bugatti Veyron Super Sport', id: 'list-03' },
{ text: 'SSC Ultimate Aero', id: 'list-04' },
{ text: 'Koenigsegg CCR', id: 'list-05' },
{ text: 'McLaren F1', id: 'list-06' },
{ text: 'Aston Martin One- 77', id: 'list-07' },
{ text: 'Jaguar XJ220', id: 'list-08' },
{ text: 'McLaren P1', id: 'list-09' },
{ text: 'Ferrari LaFerrari', id: 'list-10' }
]
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-listbox :dataSource='data'></ejs-listbox>
</div>
</template>
<script>
import { ListBoxComponent } from "@syncfusion/ej2-vue-dropdowns";
// Component registration
export default {
name: "App",
// Declaring component and its directives
components: {
"ejs-listbox": ListBoxComponent
},
// Bound properties declarations
data() {
return {
data:[
{ text: 'Hennessey Venom', id: 'list-01' },
{ text: 'Bugatti Chiron', id: 'list-02' },
{ text: 'Bugatti Veyron Super Sport', id: 'list-03' },
{ text: 'SSC Ultimate Aero', id: 'list-04' },
{ text: 'Koenigsegg CCR', id: 'list-05' },
{ text: 'McLaren F1', id: 'list-06' },
{ text: 'Aston Martin One- 77', id: 'list-07' },
{ text: 'Jaguar XJ220', id: 'list-08' },
{ text: 'McLaren P1', id: 'list-09' },
{ text: 'Ferrari LaFerrari', id: 'list-10' }
],
};
}
};
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

![vue-3-js-listbox](images/vue-3-js-listbox.PNG)

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Accessibility in Vue List box component

The List box component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the List box component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The List box component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the List box component:

| Attributes | Purpose |

| --- | --- |

| role | Indicates the List box component wrapper element as List box, the UL element as presentation, and its list item as option. |

| aria-label | Provides an accessible name for the List box component. |

| aria-multiselectable | Applied to the element with the List box role, tells assistive technologies that the list supports multiple selection. The default value is true. |

| aria-selected | Applied to elements with role option that are visually styled as selected to inform assistive technologies that the options are selected. |

Keyboard interaction

The List box component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the List box component.

| **Press** | **To do this** |

| --- | --- |

| Up arrow | Moves focus to the previous option. |

| Down arrow | Moves focus to the next option. |

| Home | Moves focus to first option. |

| End | Moves focus to last option. |

| Space | Changes the selection state of the focused option. |

| Ctrl + A | Selects all options in the list. |

| Ctrl + Shift + Home | Selects the focused option and all options up to the first option. |

| Ctrl + Shift + End | Selects the focused option and all options down to the last option. |

| Ctrl + (Up or Down) | Press Ctrl key with up / down arrow or mouse to select multiple items. |

Ensuring accessibility

The List box component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the List box component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the List box component with accessibility tools.

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs1" %}

See also

- Accessibility in Syncfusion Vue components

Data binding in Vue List box component

The ListBox loads the data either from local data sources or remote data services using the dataSourceproperty. It supports the data type of `array` or `DataManager`.

| Fields | Type | Description |

|------|------|-------------|

| text | `string` | Specifies the display text of each list item. |

| value | `string` | Specifies the hidden data value mapped to each list item that should contain a unique value. |

| groupBy | `string` | Specifies the category under which the list item has to be grouped. |

| iconCss | `string` | Specifies the iconCss class that needs to be mapped. |

| htmlAttributes | `string` | Allows additional attributes to configure the elements in various ways to meet the criteria. |

When binding complex data to the ListBox, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Local Data

Local data can be represented by the following ways as described below.

Copyright © 2001 -2024 Syncfusion Inc.

Array of string

The ListBox has support to load array of primitive data such as strings or numbers. Here, both value and text field acts as same.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data' ></ejs-listbox>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: ["Hennessey Venom", "Bugatti Chiron", "Bugatti Veyron Super
Sport", "SSC Ultimate Aero", "Koenigsegg CCR", "McLaren F1", "Aston Martin
One- 77", "Jaguar XJ220", "McLaren P1", "Ferrari LaFerrari"];
    }
  }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs2" %}

Array of object

The ListBox can generate its list items through an array of object data. For this, the appropriate columns should be mapped to the fields property.

In the following example, id and sports column from complex data have been mapped to the value field and text field, respectively.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data' ></ejs-listbox>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
```

```
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: [
            { id: 'game1', sports: 'Badminton' },
            { id: 'game2', sports: 'Cricket'},
            { id: 'game3', sports: 'Football'},
            { id: 'game4', sports: 'Golf'},
            { id: 'game5', sports: 'Tennis'},
            { id: 'game6', sports: 'Basket Ball'},
            { id: 'game7', sports: 'Base Ball'},
            { id: 'game8', sports: 'Hockey'},
            { id: 'game9', sports: 'Volley Ball'}
        ];
    fields:{ text:"sports" }
    }
  }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs3" %}

Array of complex object

The ListBox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the fields property.

In the following example, Sports.Name column from complex data have been mapped to the text field.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data' :fields="fields" ></ejs-listbox>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: [
    { Id: 'game1', Sports: { Name: 'Badminton'} },
```

```
    { Id: 'game2', Sports: { Name: 'Cricket'} },
    { Id: 'game3', Sports: { Name: 'Football'} },
    { Id: 'game4', Sports: { Name: 'Golf'} },
    { Id: 'game5', Sports: { Name: 'Tennis'} },
    { Id: 'game6', Sports: { Name: 'Basket Ball'} },
    { Id: 'game7', Sports: { Name: 'Base Ball'} },
    { Id: 'game8', Sports: { Name: 'Hockey'} },
    { Id: 'game9', Sports: { Name: 'Volley Ball'} }
];
fields:{ text:"Sports.Name" }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs4" %}

Remote Data

The ListBox supports retrieval of data from remote data services with the help of DataManager component. The Query property is used to fetch data from the database and bind it to the ListBox.

The following sample displays the first 10 products from Products table of the Northwind Data Service.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data' :query="query" :fields="fields"
></ejs-listbox>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
    data : new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor
      }),
      query : new
Query().from('Products').select('ProductID,ProductName').take(10),
      fields : { text: 'ProductName', value: 'ProductID' }
    }
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs5" %}

Drag and drop in Vue List box component

The ListBox has support to drag an item or a group of selected items and drop it within the same list box or into another list box.

The elements can be customized on drag and drop by using the following events,

| Events | Description |

|------|------|

| dragStart | Triggers when the selected element is being dragged. |

| drag | Triggers when the selected element is being dragged. |

| drop | Triggers when the selected element is being dropped. |

Single listbox

To drag and drop an item or group of item within the list box can be achieved by setting allowDragAndDrop property as true.

The following sample illustrates how to drag and drop an item within the same list box by enabling allowDragAndDrop property.

APP.VUE

```
<template>
  <div id='container' style="margin:10px auto 0; width:250px;">
      <ejs-listbox :dataSource='data' :allowDragAndDrop="true"
:fields="fields"></ejs-listbox>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: [
    { "Name": "Australia", "Code": "AU" },
    { "Name": "Bermuda", "Code": "BM" },
    { "Name": "Canada", "Code": "CA" },
    { "Name": "Cameroon", "Code": "CM" },
    { "Name": "Denmark", "Code": "DK" },
    { "Name": "France", "Code": "FR" },
    { "Name": "Finland", "Code": "FI" },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    { "Name": "Germany", "Code": "DE" },
    { "Name": "Hong Kong", "Code": "HK" }
];
fields: { text: "Name" }
    }
  }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs6" %}

Multiple listbox

To drag and drop an item or group of item between two list boxes can be achieved by setting allowDragAndDrop property as true and scope property should be set to both the list boxes.

In the following sample, the allowDragAndDrop property is set as true and scope is set as combined-list to enable drop and drop in both list boxes.

APP.VUE

```
<template>
<div id='container' style="margin:10px auto 0; width:250px;">
  <div class="listbox1">
    <h4>Group A</h4>
      <ejs-listbox id="listbox1" :dataSource='groupA'
:allowDragAndDrop="true" :fields="fields" scope="combined-list"></ejs-listbox>
  </div>
  <div class="listbox2">
    <h4>Group B</h4>
      <ejs-listbox id="listbox2" :dataSource='groupB'
:allowDragAndDrop="true" :fields="fields" scope="combined-list"></ejs-listbox>
  </div>
</div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
  return {
  groupA: [
    { "Name": "Australia", "Code": "AU" },
    { "Name": "Bermuda", "Code": "BM" },
    { "Name": "Canada", "Code": "CA" },
    { "Name": "Cameroon", "Code": "CM" },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    { "Name": "Denmark", "Code": "DK" },
    { "Name": "France", "Code": "FR" },
    { "Name": "Finland", "Code": "FI" },
    { "Name": "Germany", "Code": "DE" },
    { "Name": "Hong Kong", "Code": "HK" }
];
groupB: [
    { "Name": "India", "Code": "IN" },
    { "Name": "Italy", "Code": "IT" },
    { "Name": "Japan", "Code": "JP" },
    { "Name": "Mexico", "Code": "MX" },
    { "Name": "Norway", "Code": "NO" },
    { "Name": "Poland", "Code": "PL" },
    { "Name": "Switzerland", "Code": "CH" },
    { "Name": "United Kingdom", "Code": "GB" },
    { "Name": "United States", "Code": "US" }
];
fields: { text: "Name" }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
.listbox1{
  float: left;
  width: 48%;
}
.listbox2{
  float: right;
  width: 48%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs7" %}

Dual list box in Vue List box component

The dual list box allows the user to move items between two list boxes by clicking the toolbar buttons. Dual list box can be created by listing items in the toolbarSettings along with the scope property.

The following operations can be performed in dual list box,

| Options | Description |

|------|------------|

| moveUp | Move the selected item in the upward direction within the list box. |

| moveDown | Move the selected item in the downward direction within the list box. |

| moveTo | Move the selected item to the another list box. |

| moveFrom | Move the selected item from one list box to the another list box. |

| moveAllTo | Move all the items to the another list box. |

Copyright © 2001 -2024 Syncfusion Inc.

| moveAllFrom | Move all the items from one list box to the another list box. |

The following example illustrates how to move items from Group A to Group B list box.

APP.VUE

```
<template>
  <div id='container' style="margin:10px auto 0; width:430px;">
  <div class="listbox1">
    <h4>Group A</h4>
      <ejs-listbox :dataSource='groupA' :fields="fields" scope="#listbox"
:toolbarSettings="toolbarSettings"></ejs-listbox>
  </div>
  <div class="listbox2">
    <h4>Group B</h4>
      <ejs-listbox id="listbox" :dataSource='groupB' :fields="fields"
scope="combined-list"></ejs-listbox>
  </div>
</div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
groupA: [
    { "Name": "Australia", "Code": "AU" },
    { "Name": "Bermuda", "Code": "BM" },
    { "Name": "Canada", "Code": "CA" },
    { "Name": "Cameroon", "Code": "CM" },
    { "Name": "Denmark", "Code": "DK" },
    { "Name": "France", "Code": "FR" },
    { "Name": "Finland", "Code": "FI" },
    { "Name": "Germany", "Code": "DE" },
    { "Name": "Hong Kong", "Code": "HK" }
];
groupB: [
    { "Name": "India", "Code": "IN" },
    { "Name": "Italy", "Code": "IT" },
    { "Name": "Japan", "Code": "JP" },
    { "Name": "Mexico", "Code": "MX" },
    { "Name": "Norway", "Code": "NO" },
    { "Name": "Poland", "Code": "PL" },
    { "Name": "Switzerland", "Code": "CH" },
    { "Name": "United Kingdom", "Code": "GB" },
    { "Name": "United States", "Code": "US" }
];
fields: { text: "Name" }
toolbarSettings: { items: ['moveUp', 'moveDown', 'moveTo', 'moveFrom',
'moveAllTo', 'moveAllFrom'] }
    }
  }
}
</script>
<style>
```

```
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
.listbox1{
  float: left;
  width: 50%;
}
.listbox2{
  float: right;
  width: 44%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs8" %}

Icons and templates in Vue List box component

Icons

To place the icon on a list box, set the iconCss property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the list.

In the following sample, icon classes are mapped with iconCss field.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data' :fields="fields" ></ejs-listbox>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: [
    { text: 'Account Settings', iconCss: 'e-list-icons e-list-user-settings'
},
    { text: 'Address Book', iconCss: 'e-list-icons e-list-address-book' },
    { text: 'Delete', iconCss: 'e-list-icons e-list-delete' },
    { text: 'Forward', iconCss: 'e-list-icons e-list-forward' },
    { text: 'Reply', iconCss: 'e-list-icons e-list-reply' },
    { text: 'Reply All', iconCss: 'e-list-icons e-list-reply-all' },
    { text: 'Save All Attachments', iconCss: 'e-list-icons e-list-save-all-
attachments' },
    { text: 'Save As', iconCss: 'e-list-icons e-list-icon-save-as' },
    { text: 'Touch/Mouse Mode', iconCss: 'e-list-icons e-list-touch' },
    { text: 'Undo', iconCss: ' e-list-icons e-list-undo' }
];
fields: { text: "text", iconCss: "iconCss" }
    }
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@font-face {
  font-family: 'e-listbox-icons';
  src:
  url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tSfsAAAEoAAAAVmNtYXDnKOeOAAABrAAAAEhnbHlmwmW
IZQAAAgwAAA78aGVhZBSPflEAAADQAAAANmhoZWEIUQQMAAAArAAAACRobXR4LAAAAAAAAYAAAAA
sbG9jYRUAGIYAAAH0AAAAGG1heHABGwGxAAABCAAAACBuYW1lD1KnmAAAEQgAAAKFcG9zdNYJPxo
AABOQAAAArgABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAACwABAAAAAQAATvHvyF8
PPPUACwQAAAAAANi+nLAAAAAA2L6csAAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAALAaUABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnCQQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAAFAA
EADQAAAAEAAQAAQAA5wn//wAA5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcACAAJAAoAAAAAAdA
CDAI4AwADhAVmBnQGlgcAB34AAwAAAAADLgPuAK8BMwGZAAABER0BHwU7AT8FPQE/BjsBHwYdAR8
GPwY9AT8HHwYdAR8HPwc9AT8GHwYdAQ8LIy8PNT8HFR8HPwcRNT8GHwYHEQ8NFR8RPxYvDisBDwI
vBisBDwIvByMHPQEvDSsBDw0nFR8GMz8ELwQ1Pw07AR8ODwQfBDsBPwYvDw8OAaACAwQEBgYGBgY
FBQQDAgEBAgMDAwMEDgQDAwICAQECAwQFBQYGBgYGBAQDAgECAQMDAwMEDgQDAwMCAQECAgQFBQY
GBwYFBQMDAgEBAgMDAwQEDgMEAwICAQoKCgcGBwkLCwwN6AoJCQkICAcGDw0LCg0GBQIBAQIEBAY
EBQUBAgMDBQUGBwYGBQUEAgIBAQIDAwQFBgUFBAQDAwFwCwkHBwcGBgoIBgcFAwIBAgMECAgUDQ8
SCwwNDQ4PEA/nDAsMCwoKCgkJCAcHBgYEBAYKBwgFAgEBAgMDBAUFBgYHBwgICAgSBwcKBgcICQk
KCwsSCAgLBwcHCQgKCQoTCwICAwQEBQYGBwcICAkICQkJCAgIBwcGBQUEBAMCAVoCBAcGAgUGAwM
CAwMBAQcFAgEDAwUGBwgJCgsLDQ0NDg4PDg0NDAwLCgkIBwYFAwIBAQMEBwEBAwIDAwQEBAMDBwY
DAQECBQUICAoLDA4ODxAQERIREREPEA4NDAsKCQcGBAMC5P7+TAYGBgQEAwICAwQEBgYGTAUDAwM
CAgEBAgMDAwQEZwYGBQUEAgIBAQICBAUFBgZLBgQDAgMBAQEBAQICAwQEBFsGBgUFBAMBAQEBAwQ
FBQYGOQQDBAMCAgEBAQECAgMEAwRKWDMlGQ0LCQgHBQIJAQIDBAUGBxESERIeFRgNDg4ODw4ICQQ
DAkYGBgUFBAMBAQEBAwQFBQYGAb8FBQUDBAICAQEBAwMEBAUG/sUBAQMDBAQFCgsLEBMTExISERA
PGhUnFRUVCwkJBwUEAwEJAQICBAQGBgYICAgJCgsKDBQoJDAuYAgJBwgHBwYGBQUEBAMCAQECAwo
IBwUFBAIBAgQJBwUFBAMBAQKyCggJCAgHBwYGBQQEAwICAgIDAwUFBgYHBwgICAlVChITEgwCAwE
BAgIFBgMSDw8QDg4NDQ0LCwoJCAcGBQMDAwMFBgcICQoLCw0NDQ4OEA8PEgMGBQICAQIDAxMSExM
REREQEQw4NDAsKCQcGBAMBAQMEBgcJCgsMDQ4PEBERAAAAAAUAAAAAA/QDgQAHABEAFQAbACIAABM
HFzUzNSM1FzMVIxUhEQ8BAR8BASEHNzUhNSEnETcRITUhc2fP7u5etrYCu6O5/qHFlwEz/ZqdNQI
4/ZNnNQI3/ZQBmF68d4d3TtUjAcqgswFTZJoBM5YvoDUy/m8vAWI2AAMAAAAAA/QDHQAIABIAFwA
AEwcfATUhNSE1FzMVIxUhEQ8BAR8BPwEhfHB0cAEE/vxR0tICs6a1/qjCmZaZ/Z8CEWdhZ4CTgGf
IIgHFoLIBUmSZmZkAAAAGAAAAAAOiA/MAaQBtAI4AkwCXAJ0AAAETMwM/CR8IEw8GLwQRPwQfBjM
vCA8GEx8HPwgDLwoPCQEzNSMDITUhNSE1Pw8fAxEjFSE1IyU7ATUjBTM1IwMzETM1IwMDAxYDAQQ
CAwUFBggJCgkOCgUEBAQCAQMBBgUHBQQGDQgJBwcDBQMEBAkDAwIBAQMZAwIEBQgEBQYHCgYGBgU
EAwMIBgkGBgcICRAKCgUFBQUDBAMDBAQEBQcICQsMDw4LDAYGBgUFBAP+gFhYsgIT/mEBogEBAwQ
EBQYHCAgJCQoKCwsMDAwLd/47dAHFSyl0/nlYWLVLKXQBr/7dARwRCQUFBQQEAwMCAQQFAwUFBgc
I/pIFCwYFAgEBAQIGBwkBKQYHAwIBAQICAwIG9/cICQcFAgIBAQEBAwQGCAz+0RIJCAQEAwICAgM
GBAUGBwgJAW4QDAYHBgcFBQQCAgICBQQEBgcHCgoBALP9UFhETgsLCgoJCQgIBwYFBAQCAgEBAgM
EAWjV1ShbW1v9SgJbWwAABQAAAAADtQP0AAMABwAzAFcAZQAAARUhNQEVIzUjFR8GOwI/BT0BFxE
jPQEvBiEPBxUjEScRFR8GIT8HES8HIQ8GJxEzESE1IQ8GAtv+5wEZnT4BAQMEBQUGBtsGBgYEBAM
CXl4CAwQEBgYG/qgGBgUFBAMBAV4+AgMEBAYGBgKQBwUGBAQDAgEBAgMEnwUHA/4IBgYGBAQDAp0
/AbX+LAcFBgQEAwIBBru7AhN9fZwGBgUFBAMCAgMEBQUGBnBe/bzaBwUGBAQDAgEBAgMEBAYFB9o
CziD88gYGBQUEAwEBAQEDBAUFBgYCcQYGBgWeBAIBAQIDAwUFBpb9EgLPPgEBAwQFBQYABQAAAAA
D9AP0ACsAbAC7AQMBpAAAARUfCDsBPwg9AS8JDwkXFQ8PLw8/Dx8OJTsCHwcPAScHFw8EJwcXFR8
DByEjLw09AT8dJQ8SKwEvEj8PHw4FHw8PEBUfDyEXNzM1Nx8EBxc3HwE/ARc3Jz8DFzcnPwM1Nyc
HLwQ3JwcvAzUjFS8HPw49AS8PKwEPDwKcAQIFBgkKCwYGBwYGBgsKCQcEAgEBAgQHCQoLBgYGBwY
GCwoJBgUCAdsBAgQFBwgJCgsMDQ4ODw8QEBAPDg4NDAsKCQgGBgQCAQECBAYGCAkKCwwNDg4PEBA
PEA8ODg0MCwoJCAcFBAL94UtLExMTEhISERAKEQ8pLygJDAsJBD4LPwIEBgUQ/pcIBwgHBwYGBgU
```

```
EBAMDAgEBAgIDBAQFBQYHBwcICAkKCQsKCwsMDAwNDA0NDg4BBgECAwQFBgcICAoKCwwNDQ0MDAw
MDAwMCw4NDAwLCwkJCAcGBQQDAQEBAgUGBwkKCwwNDw8QEBIREhEREA8ODQ0LCgkHBgQD/mcBAQI
CAwQEBQUGBwcHCQgPFRoYGBcVFBIQDw0LCgcEAgEBAgMFBgcICQoLCw0NDQ4OAVYJHAEdDw8IEQ0
YOxgYEhQXGDoXFBAPEDogOQYFBAI/Cz4ECAoMCykwKRYTFBc+Eg4PDw8QEBcPCQgIBwYGBgUEBAM
CAgEBBAYICQwODxESExUWFhgMDAwMFxcVFRMTEBANDAoIBgMBASUGBgYMCggHBAIBAQIEBwgKDAY
GBgcGBgsKCAcFAQEBAQEBBQcICgsGBgcICA8PDg4NDAsKCQgHBQQCAQECBAUHCAkKCwwNDQ8PDxA
QEA8ODg0MCwoJCAYFBAMBAQMEBQYICQoLDA0ODg8Q6gIEBAYHCQkHCAgxKDELERMUDAs/Cw4VFBM
OCQECAwMEBAUGBQcHBwcIBw4ODQ0NDQwMDAwLCwoKCgkJCAgIBwYGBQUEBAMCAgHrDw4NDgwNDAs
KCgkJBwcFBQMCAQECAwUFBwcICgkLCwwNDA4NDg8REREPDw8NDAsKCQcGBAMBAQMEBgcJCgsMDQ8
PDxEREQ0NDA0MDAsMCwoLCgkKCQ0FCQsNDxATExUWGBkZGxsODg8ODg0NDAwKCgkIBwYFAwIBEBA
BERAOBwsHQhZDBAEBBEMWQQwMDRIiNyEOExUVDgs+CwwTEhENMSkxDAgGBUAoDQkJCAcGBQYOCAo
JCgsKDAsMCwwNDA0NDAwXFhYUFBIQDw4MCgcGBAEBBAYHCgwODxASFBQWFhgLAAIAAAAAA78D8wA
DAOwAAAEVITUlDwcdAR8HPwcvATU/BxUjDw4VHwg/BjU/ChUPDxUfBzsBPxEVDw8VHwc7AT8RFQ8
QHwc7AT8RFSERISMPBQMQ/mH+/QYGBAUGBAMBAgIDBAQFBQUFBQQEBAICAQIBAQIEBQoLCwwMDAs
LCwoJCQYFBQQHBAMBAQECBAMFBQUFBgUEBAMDAQMDBQMEBAUFCwsMDAwLCwsKCQkGBQUEBgUCAgE
BAwMEBAUFBgUFBAQEAgEBAgMFAwQEBQULCwwMCwwLCgoKCAcFBQQGBQICAQEDAwQEBQUGBQUEBAQ
CAQECAwUDBAQFBQsLDAwLDAsKCgoIBgYFBAYFAgEBAQEDAwQEBQUGBQUEBAQCAQECAwUDBAQFBQs
LDAL6/QYMCwwLCwoLA43e3ksHBgcIDhAQERERBgUFAwMCAQEBAQIDAwUFBgsYDAwLCQgGBAIBmQE
CAwQFBgcHBwYIDhAQERERBgUFAwMCAQEBAQIDAwUFKhILCgQFAwMDBAIBnAEBAQMEBQYHBwcHBw8
PEBEREQYGBAQCAgICAgIEBAYqEgsKBAQEAwMEAgGWAQEBAwQFBgcHBwcHDw8QERERBgYEBAICAgI
CAgQEBioSCwoEBAQDAwQCAZYBAQEDBAUGBwcHBwcPDxAREREGBQUEAgICAgICBAUFKhILCgQEBAM
DBAIBbgPYAQIDBAUGAAAAAAAEAAAAAA/QD9AALAAATCQEXCQE3CQEnCQEMATL+zsIBMgEywv7OATL
C/s7+zgMy/s7+zsIBMv7OwgEyATLC/s4BMgAAAwAAAAD9AMyADQATABQAAABIw8aPw4zHwQVNyc
BIT8SNQkBHwEBIQMvEhIiIB4cGxgXFRQSEQ8ODAwKCQgHBgYEBwQDAQkKCw0NDw8QERIREhISExI
jIh8nK8XF/N0BTggODA0PERMVGBodIBESExMUFRb+ov6ixZwBMv2WAecBAgMEBgYHCAkKCgsLCww
MDAwNDAwLDBUTEREQEA0NCgoICAYFBAMCAgEBAwMHCV6vyP7tERoREhISERAQDg4MBQUFBAMDA+T
+qAFYZ5wBOAAAAAABAAAAAAP0A68AbgAAExczJyEfExUPDxUzPx09AS8dIyE3JyMMztOWAWIYGBc
VFRQSEQ8ODAUFBAQDAwIBAQIDBQYICgsNDhASEhUVGDU9EhMSERERERA8PDw4NDQwLCwoKCQkHBwc
FBQQEAwECAgIEBAUGBwgICQoLCwwMDg0PDw8QEBEREhISExMTE/6mkgLRArXzsQEDBAYHCQoMDg8
QCQoJCgoLCwwLGRcWFBMREA4NCwoIBwQEAQGNAQIDAwUFBQcHBwkJCQsLCwwNDQ4ODxAQEBESEhI
TExQTEhISERAQEA8PDQ4NDAwLCgoJCAgHBwUFBAQDAgGrAgAAABIA3gABAAAAAAAAAAAEAAAABAAA
AAAABAA8AAQABAAAAAAACAAcAEAABAAAAAAADAA8AFwABAAAAAAAEAA8AJgABAAAAAAAFAAsANQA
BAAAAAAAGAA8AQAABAAAAAAAKACwATwABAAAAAAALABIAewADAAEECQAAAAIAjQADAAEECQABAB4
AjwADAAEECQACAA4ArQADAAEECQADAB4AuwADAAEECQAEAB4A2QADAAEECQAFABYA9wADAAEECQA
GAB4BDQADAAEECQAKAFgBKwADAAEECQALACQBgyBlLWxpc3Rib3gtaWNvbnNSZWd1bGFyZS1saXN
0Ym94LWljb25zZS1saXN0Ym94LWljb25zVmVyc2lvbiAxLjBlLWxpc3Rib3gtaWNvbnNGb250IGd
lbmVyYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQA
gAGUALQBsAGkAcwB0AGIAbwB4AC0AaQBjAG8AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGwAaQBzAHQ
AYgBvAHgALQBpAGMAbwBuAHMAZQAtAGwAaQBzAHQAYgBvAHgALQBpAGMAbwBuAHMAVgBlAHIAcwB
pAG8AbgAgADEALgAwAGUALQBsAGkAcwB0AGIAbwB4AC0AaQBjAG8AbgBzAEYAbwBuAHQAIABnAGU
AbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAB
yAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAA
CAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAsBAgEDAQQBBQEGAQcBCAEJAQoBCwEMAAh
0b3VjaC13ZglyZXBseS1hbGwFcmVwbHkUc2F2ZS1hbGwtYXR0YWNobWVudHMKc2F2ZS1hcy13ZhN
1c2VyLXNldHRpbmdzLTAyLXdmDmFkcmVzcy1ib29rLTAzBmRlbGV0ZQdmb3J3YXJkB3VuZG9fMDE
AAAAA) format('truetype');
  font-weight: normal;
  font-style: normal;
}

.e-list-icons {
  font-family: 'e-listbox-icons';
  speak: none;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
```

```
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.e-list-touch:before {
  content: "\e700";
}
.e-list-reply-all:before {
  content: "\e701";
}
.e-list-reply:before {
  content: "\e702";
}
.e-list-save-all-attachments:before {
  content: "\e703";
}
.e-list-icon-save-as:before {
  content: "\e704";
}
.e-list-user-settings:before {
  content: "\e705";
}
.e-list-address-book:before {
  content: "\e706";
}
.e-list-delete:before {
  content: "\e707";
}
.e-list-forward:before {
  content: "\e708";
}
.e-list-undo:before {
  content: "\e709";
}
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs15" %}

Templates

ListBox items can be customized according to the requirement using itemTemplate property.

In the following sample, the items in the cart are displayed using list box template,

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data' :itemTemplate= "itemtemplate"
></ejs-listbox>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  data (){
    return {
    itemtemplate: "<div class='list-wrapper'><span class='${pic} e-avatar e-avatar-xlarge e-avatar-circle'></span><span class='text'>${text}</span><span class='description'>${description}</span></div>",
       data: [
    { text: 'JavaScript', pic: 'javascript', description: 'It is a lightweight interpreted or JIT-compiled programming language.' },
    { text: 'TypeScript', pic: 'typeScript', description: 'It is a typed superset of Javascript that compiles to plain JavaScript.' },
    { text: 'Angular', pic: 'angular', description: 'It is a TypeScript-based open-source web application framework.' },
    { text: 'React', pic: 'react', description: 'A JavaScript library for building user interfaces. It can also render on the server using Node.' },
    { text: 'Vue', pic: 'vue', description: 'A progressive framework for building user interfaces. it is incrementally adoptable.' }
];
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
.e-listbox-wrapper {
  margin: auto;
  max-width: 400px;
  box-sizing: border-box;
}
#container {
  box-sizing: border-box;
}
#container .e-listbox-wrapper:not(.e-list-template) .e-list-item {
  padding: 0;
  position: unset;
  cursor: pointer;
  height: 76px;
  line-height: normal;
}
.list-wrapper {
    height: inherit;
    position: relative;
    padding: 14px 12px 14px 78px;
}
.list-wrapper .text,
.list-wrapper .description {
  display: block;
  margin: 0;
  padding-bottom: 3px;
  white-space: normal;
}
.list-wrapper .description {
  font-size: 12px;
  font-weight: 500;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
#container .e-listbox-wrapper .list-wrapper .text {
  font-weight: bold;
  font-size: 13px;
}
.list-wrapper .e-avatar {
    position: absolute;
    left: 5px;
    background-color: transparent;
    font-size: 22px;
    top: calc(50% - 33px);
}
.javascript {
  background-image: url('jses5-1.svg');
  width: 70px;
  height: 60px;
  background-repeat: no-repeat;
}
.typeScript {
  background-image: url('javascript1.svg');
  width: 70px;
  height: 60px;
  background-repeat: no-repeat;
}
.angular {
  background-image: url('angular1.svg');
  width: 70px;
  height: 60px;
  background-repeat: no-repeat;
}
.vue {
  background-image: url('vue.svg');
  width: 70px;
  height: 60px;
  background-repeat: no-repeat;
}
.react {
  background-image: url('react.svg');
  width: 70px;
  height: 60px;
  background-repeat: no-repeat;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs16" %}

Selection in Vue List box component

The ListBox provides support to select an item or a group of item by mouse or keyboard action. There are two selection modes available in list box,

- Single - To select single item in the list box.
- Multiple - To select multiple items in the list box.

On selection of each list box item, change event is triggered.

Copyright © 2001 -2024 Syncfusion Inc.

Single selection

To enable single selection in the list box, mode should be set as Single in selectionSettings property.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data'
:selectionSettings="selectionSettings" ></ejs-listbox>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
];
selectionSettings: { mode: "Single" }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs17" %}

Multiple selection

To enable multiple selection in the list box, mode should be set as Multiple in selectionSettings property.

To select multiple items, use the SHIFT, CTRL, and arrow keys to make selections.

By default, the selection mode is set as Multiple.

APP.VUE

```
<template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data'
:selectionSettings="selectionSettings" ></ejs-listbox>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
];
selectionSettings: { mode: "Multiple" }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs18" %}

Sorting and grouping in Vue List box component

Sorting

The ListBox supports sorting of available items in the alphabetical order that can be either ascending or descending. This can achieved using sortOrder property. Sort order can be `None`, `Ascending` or `Descending`.

In the following example, the `SortOrder` is set as `Descending`.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='groupA' sortOrder="Descending"
:fields="fields" ></ejs-listbox>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
groupA: [
    { "Name": "Australia", "Code": "AU" },
    { "Name": "Bermuda", "Code": "BM" },
    { "Name": "Canada", "Code": "CA" },
    { "Name": "Cameroon", "Code": "CM" },
    { "Name": "Denmark", "Code": "DK" },
    { "Name": "France", "Code": "FR" },
    { "Name": "Finland", "Code": "FI" },
    { "Name": "Germany", "Code": "DE" },
    { "Name": "Hong Kong", "Code": "HK" }
];
fields: { text:"Name"}
    }
  }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs19" %}

Grouping

To get start quickly about how to Link to the Video the Vue ListBox items, you can check on this video:

The ListBox supports to wrap the nested element into a group based on its category. The category of each list item can be mapped with groupBy field in the data table.

In the following example, vegetables are grouped based on its category.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data' :fields="fields" ></ejs-listbox>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
```

```
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
data: [
    { "Vegetable": "Cabbage", "Category": "Leafy and Salad", "Id": "item1"
},
    { "Vegetable": "Spinach", "Category": "Leafy and Salad", "Id": "item2"
},
    { "Vegetable": "Wheat grass", "Category": "Leafy and Salad", "Id":
"item3" },
    { "Vegetable": "Yarrow", "Category": "Leafy and Salad", "Id": "item4" },
    { "Vegetable": "Pumpkins", "Category": "Leafy and Salad", "Id": "item5"
},
    { "Vegetable": "Chickpea", "Category": "Beans", "Id": "item6" },
    { "Vegetable": "Green bean", "Category": "Beans", "Id": "item7" },
    { "Vegetable": "Horse gram", "Category": "Beans", "Id": "item8" },
    { "Vegetable": "Garlic", "Category": "Bulb and Stem", "Id": "item9" },
    { "Vegetable": "Nopal", "Category": "Bulb and Stem", "Id": "item10" },
    { "Vegetable": "Onion", "Category": "Bulb and Stem", "Id": "item11" }
];
fields: { groupBy:"Category", text:"Vegetable", value:"Id" }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs20" %}

Style and appearance in Vue List box component

To modify the ListBox appearance, you need to override the default CSS of ListBox component. Please find the list of CSS classes and its corresponding section in ListBox component. Also, you have an option to create your own custom theme for the controls using our Theme Studio.

CSS Class | Purpose of Class

|.e-listbox-wrapper|To customize the listbox wrapper

|.e-list-parent .e-list-item|To customize the listbox list items

|.e-list-parent .e-list-item:hover|To customize the listbox list items on hover

|.e-list-parent .e-list-item.e-selected|To customize the listbox selected list item

|.e-listboxtool-wrapper .e-listbox-tool|To customize the listbox toolbar

|.e-listboxtool-wrapper .e-listbox-tool .e-btn|To customize the listbox toolbar button

|.e-listboxtool-wrapper .e-listbox-tool .e-btn .e-btn-icon.e-icons::before|To customize the listbox toolbar icon

Copyright © 2001 -2024 Syncfusion Inc.

Horizontal ListBox

You can use cssClass property to display the Listbox horizontally.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data' cssClass='e-horizontal-listbox'
></ejs-listbox>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
];
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
/* Custom css for horizontal listbox */
.e-horizontal-listbox .e-list-parent {
  display: inline-flex;
  align-items: center;
}
.e-horizontal-listbox {
  overflow-y: hidden;
  height: 100px;
}
.e-horizontal-listbox .e-list-parent .e-list-item {
  width: max-content;
  line-height: 100px;
  height: 100px;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs21" %}

How To

Add items in Vue List box component

To add an item or multiple items, addItems method can be used. In the following example, the Bugatti Veyron Super Sport and SSC Ultimate Aero items will be added while clicking Add Items button.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox id="listbox" :dataSource='data'  ></ejs-listbox>
        <button id="additem" class="e-btn" v-on:click="additem">Add
Items</button>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
];
    }
  },
  methods:{
    additem: function(e){
        var items = ["Bugatti Veyron Super Sport", "SSC Ultimate Aero"]
        var listboxobj =
document.getElementById("listbox").ej2_instances[0];
        if (!listboxobj.getDataByValue('Bugatti Veyron Super Sport')) {
        listboxobj.addItems([{ text: 'Bugatti Veyron Super Sport', id:
'list-03' }, { text: 'SSC Ultimate Aero', id: 'list-04' }]);
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs10" %}

Enable or disable items in Vue List box component

To enable or disable items in the list box, enableItems method can be used. In the following example, the Bugatti Veyron Super Sport and SSC Ultimate Aero items are disabled by default and by clicking Enable Items buttons, the disabled items will be enabled.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox id="listbox" :dataSource='data' :created="onCreated"
></ejs-listbox>
        <button id="enableitem" class="e-btn" v-on:click="enableitem">Enable
Items</button>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
];
    }
  },
  methods:{
  onCreated: function(e){
    var listboxobj= document.getElementById("listbox").ej2_instances[0];
    listboxobj.enableItems(["Bugatti Veyron Super Sport", "SSC Ultimate
Aero"],false)
}
  enableitem: function(e){
    var listboxobj1= document.getElementById("listbox").ej2_instances[0];
    listboxobj1.enableItems(["Bugatti Veyron Super Sport", "SSC Ultimate
Aero"],true)
  }
}
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs11" %}

Enable scroller in Vue List box component

The ListBox supports scrolling and it can be achieved by restricting the height of the list box using height property.

In the following sample, height of the list box is restricted to 290px.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox :dataSource='data' height="290px" ></ejs-listbox>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
];
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs12" %}

Form submit in Vue List box component

In the following code snippet, the value that is in selected state will be sent on form submit.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
    <form>
        <ejs-listbox :dataSource='data' ></ejs-listbox>
        <button class="e-btn e-primary">Submit</button>
    </form>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
];
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs13" %}

Select items in Vue List box component

In the following example, Bugatti Chiron is selected using selectItems method.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:10px auto 0; width:250px;">
        <ejs-listbox id="listbox" :dataSource='data' :created="onCreated"
></ejs-listbox>
    </div>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { ListBoxPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(ListBoxPlugin);
export default {
  data (){
    return {
       data: [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
];
    }
  },
  methods:{
      onCreated: function(e){
        var listboxobj= document.getElementById('listbox').ej2_instances[0];
        listboxobj.selectItems(['Bugatti Chiron']);
}
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/list-box/getting-started/getting-started-cs14" %}

ListView

Getting Started with the Vue Listview Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Listview component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

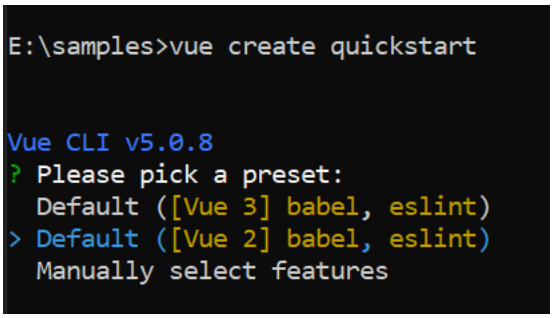
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Listview component as an example. Install the @syncfusion/ej2-vue-lists package by running the following command:

`bash

npm install @syncfusion/ej2-vue-lists --save

`

or

`bash

yarn add @syncfusion/ej2-vue-lists

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

Copyright © 2001 -2024 Syncfusion Inc.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Listview component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-lists/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Listview component using Composition API or Options API:

1\. First, import and register the Listview component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { ListViewComponent as EjsListview } from '@syncfusion/ej2-vue-
lists';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { ListViewComponent } from '@syncfusion/ej2-vue-lists';
export default {
components: {
'ejs-listview': ListViewComponent
}
}
</script>
```

2\. In the template section, define the Listview component with the dataSource property.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-listview id='flat-list' :dataSource='dataSource' width="330px"></ejs-
listview>
</div>
</template>
```

Bind data source

Populate the data in ListView by using the dataSource property. Here, an array of JSON values are passed to the ListView component as dataSource in data option inside the <script> section.

COMPOSITION API (~/SRC/APP.VUE)

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<div id="app">
<ejs-listview id='flat-list' :dataSource='dataSource' width="330px"></ejs-
listview>
</div>
</template>
<script setup>
import { ListViewComponent as EjsListview } from '@syncfusion/ej2-vue-
lists';
const dataSource = [
{ text: 'Hennessey Venom', id: 'list-01' },
{ text: 'Bugatti Chiron', id: 'list-02' },
{ text: 'Bugatti Veyron Super Sport', id: 'list-03' },
{ text: 'SSC Ultimate Aero', id: 'list-04' },
{ text: 'Koenigsegg CCR', id: 'list-05' },
{ text: 'McLaren F1', id: 'list-06' },
{ text: 'Aston Martin One- 77', id: 'list-07' },
{ text: 'Jaguar XJ220', id: 'list-08' },
{ text: 'McLaren P1', id: 'list-09' },
{ text: 'Ferrari LaFerrari', id: 'list-10' },
];
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-listview id='flat-list' :dataSource='dataSource' width="330px"></ejs-
listview>
</div>
</template>
<script>
import { ListViewComponent } from '@syncfusion/ej2-vue-lists';
export default {
components: {
'ejs-listview': ListViewComponent
},
name: 'app',
data () {
return {
dataSource : [
{ text: 'Hennessey Venom', id: 'list-01' },
{ text: 'Bugatti Chiron', id: 'list-02' },
{ text: 'Bugatti Veyron Super Sport', id: 'list-03' },
{ text: 'SSC Ultimate Aero', id: 'list-04' },
{ text: 'Koenigsegg CCR', id: 'list-05' },
{ text: 'McLaren F1', id: 'list-06' },
{ text: 'Aston Martin One- 77', id: 'list-07' },
{ text: 'Jaguar XJ220', id: 'list-08' },
{ text: 'McLaren P1', id: 'list-09' },
{ text: 'Ferrari LaFerrari', id: 'list-10' },
]
}
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSTION API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
    <ejs-listview id='flat-list' :dataSource='dataSource'
width="330px"></ejs-listview>
  </div>
</template>
<script setup>
import { ListViewComponent as EjsListview } from '@syncfusion/ej2-vue-
lists';
const dataSource = [
        { text: 'Hennessey Venom', id: 'list-01' },
        { text: 'Bugatti Chiron', id: 'list-02' },
        { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
        { text: 'SSC Ultimate Aero', id: 'list-04' },
        { text: 'Koenigsegg CCR', id: 'list-05' },
        { text: 'McLaren F1', id: 'list-06' },
        { text: 'Aston Martin One- 77', id: 'list-07' },
        { text: 'Jaguar XJ220', id: 'list-08' },
        { text: 'McLaren P1', id: 'list-09' },
        { text: 'Ferrari LaFerrari', id: 'list-10' },
      ];
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-vue-lists/styles/material.css";
  #flat-list {
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
  #app {
    color: #008cff;
    height: 40px;
    left: 30%;
    position: absolute;
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
    <ejs-listview id='flat-list' :dataSource='dataSource'
width="330px"></ejs-listview>
  </div>
</template>
<script>
import { ListViewComponent } from '@syncfusion/ej2-vue-lists';
export default {
  components: {
    'ejs-listview': ListViewComponent
  },
  name: 'app',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    data () {
     return {
        dataSource : [
         { text: 'Hennessey Venom', id: 'list-01' },
         { text: 'Bugatti Chiron', id: 'list-02' },
         { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
         { text: 'SSC Ultimate Aero', id: 'list-04' },
         { text: 'Koenigsegg CCR', id: 'list-05' },
         { text: 'McLaren F1', id: 'list-06' },
         { text: 'Aston Martin One- 77', id: 'list-07' },
         { text: 'Jaguar XJ220', id: 'list-08' },
         { text: 'McLaren P1', id: 'list-09' },
         { text: 'Ferrari LaFerrari', id: 'list-10' },
       ]
     }
   }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-vue-lists/styles/material.css";
  #flat-list {
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
  #app {
    color: #008cff;
    height: 40px;
    left: 30%;
    position: absolute;
  }
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/listview/getting-started-cs1" %}

See Also

Data Binding Types

ListView customization

Virtualization

Copyright © 2001 -2024 Syncfusion Inc.

Getting Started with the Vue ListView Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue ListView component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Copyright © 2001 -2024 Syncfusion Inc.

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue ListView component as an example. To use the Vue ListView component in the project, the @syncfusion/ej2-vue-lists package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-lists --save

`

or

`bash

yarn add @syncfusion/ej2-vue-lists

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the ListView component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-lists/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue ListView component using Composition API or Options API:

1.First, import and register the ListView component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { ListViewComponent as EjsListview } from '@syncfusion/ej2-vue-
lists';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { ListViewComponent } from '@syncfusion/ej2-vue-lists';
//Component registration
export default {
name: "App",
components: {
"ejs-listview": ListViewComponent
}
}
</script>
```

2.Add the component definition in template section.

~/SRC/APP.VUE

```
<template>
<ejs-listview id='sample-list-flat' :dataSource='data'
showCheckBox='true'></ejs-listview>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-listview id='sample-list-flat' :dataSource='data'
showCheckBox='true'></ejs-listview>
</template>
<script setup>
import { ListViewComponent as EjsListview } from '@syncfusion/ej2-vue-
lists';
const data = [
{ text: 'Hennessey Venom', id: 'list-01' },
{ text: 'Bugatti Chiron', id: 'list-02' },
{ text: 'Bugatti Veyron Super Sport', id: 'list-03' },
{ text: 'SSC Ultimate Aero', id: 'list-04' },
{ text: 'Koenigsegg CCR', id: 'list-05' },
{ text: 'McLaren F1', id: 'list-06' },
{ text: 'Aston Martin One- 77', id: 'list-07' },
{ text: 'Jaguar XJ220', id: 'list-08' },
{ text: 'McLaren P1', id: 'list-09' },
{ text: 'Ferrari LaFerrari', id: 'list-10' },
];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-listview id='sample-list-flat' :dataSource='data'
showCheckBox='true'></ejs-listview>
</template>
<script>
import { ListViewComponent } from '@syncfusion/ej2-vue-lists';
//Component registration
export default {
name: "App",
components: {
"ejs-listview": ListViewComponent
},
data() {
return {
data: [
{ text: 'Hennessey Venom', id: 'list-01' },
{ text: 'Bugatti Chiron', id: 'list-02' },
{ text: 'Bugatti Veyron Super Sport', id: 'list-03' },
{ text: 'SSC Ultimate Aero', id: 'list-04' },
{ text: 'Koenigsegg CCR', id: 'list-05' },
{ text: 'McLaren F1', id: 'list-06' },
{ text: 'Aston Martin One- 77', id: 'list-07' },
{ text: 'Jaguar XJ220', id: 'list-08' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ text: 'McLaren P1', id: 'list-09' },
{ text: 'Ferrari LaFerrari', id: 'list-10' },
],
};
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

Copyright © 2001 -2024 Syncfusion Inc.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Data binding in Vue Listview component

ListView provides an option to load the data either from local dataSource or remote data services.

This can be done through the dataSource property that supports the data type of array or DataManager.

ListView supports different kind of data services such as OData, OData V4, and Web API, and

data formats like XML, JSON, and, JSONP with the help of DataManager Adaptors.

| Fields | Type | Description |

|------|------|-------------|

| id | string | Specifies ID attribute of list item, mapped in dataSource. |

Copyright © 2001 -2024 Syncfusion Inc.

| text | string | Specifies list item display text field. |

| isChecked | string | Specifies checked status of list item. |

| isVisible | string | Specifies visibility state of list item. |

| enabled | string | Specifies enabled state of list item. |

| iconCss | string | Specifies the icon class of each list item that will be added before to the list item text. |

| child | string | Specifies child dataSource fields. |

| tooltip | string | Specifies tooltip title text field. |

| groupBy | string | Specifies category of each list item. |

| sortBy | string | Specifies sorting field, that is used to sort the listview data. |

| htmlAttributes | string | Specifies list item html attributes field. |

When complex data bind to ListView, you should map the fields properly. Otherwise, the ListView properties remain as undefined or null.

Bind to local data

Local data can be represented in two ways, they are as follows:

- Array of simple data.
- Array of JSON data.

Array of simple data

ListView supports to load the array of primitive data like string and numbers. Here, both value and text field act as same.

APP.VUE

```
<template>
  <div class="control-section">
    <div id = 'flat-list'>
    <!-- ListView element -->
    <ejs-listview id='sample-list-flat' :dataSource='arts'></ejs-listview>
    </div>
  </div>
</template>
<style>
#sample-list-flat {
    border: 1px solid #dddddd;
    border-radius: 3px;
    margin: auto;
}
#flat-list {
    width: 50%;
    padding: 10px;
    margin: auto;
}
</style>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
export default {
  data: function() {
    return {
      arts: ["Artwork", "Abstract", "Modern Painting", "Ceramics",
"Animation Art", "Oil Painting"];
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/data-binding/simple-data-cs1" %}

Array of JSON data

ListView can generate its list items through an array of complex data. To get it work properly, you should map the appropriate columns to the field property.

In the following example, role column is mapped with the text field.

APP.VUE

```
<template>
  <div class="control-section">
    <div id = 'flat-list'>
    <!-- ListView element -->
    <ejs-listview id='sample-list-flat' :dataSource='settings'
:fields='fields' :headerTitle='headerTitle' showHeader='true'></ejs-
listview>
    </div>
  </div>
</template>
<style>
#sample-list-flat {
    border: 1px solid #dddddd;
    border-radius: 3px;
    margin: auto;
}
#flat-list {
    width: 50%;
    padding: 10px;
    margin: auto;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
export default {
  data: function() {
    return {
      settings: [
    {
        'Name': 'Display',
        'id': 'list-01'
    },
    {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        'Name': 'Notification',
        'id': 'list-02'
    },
    {
        'Name': 'Sound',
        'id': 'list-03'
    },
    {
        'Name': 'Apps',
        'id': 'list-04'
    },
    {
        'Name': 'Storage',
        'id': 'list-05'
    },
    {
        'Name': 'Battery',
        'id': 'list-06'
    }
],
fields: { text: 'Name', tooltip: 'Name', id:'id'},
    //set the header tittle for the list
    headerTitle: 'Device settings',
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/data-binding/array-data-cs1" %}

Bind to remote data

The ListView supports to retrieve the data from remote data services with the help of DataManager component.

The Query property allows to fetch data and return it to the ListView from the database.

In the following sample, first 10 Employees from the ListView table are displayed.

APP.VUE

```
<template>
  <div class="control-section">
  <div id="flat-list">
    <!-- ListView element -->
    <ejs-listview id='sample-list' :dataSource='data' :query='query'
:fields='fields' :headerTitle='headerTitle' showHeader='true'></ejs-
listview>
    </div>
</div>
</template>
<style>
#sample-list {
    border: 1px solid #dddddd;
    border-radius: 3px;
    margin: auto;
}
#flat-list {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    width: 50%;
    padding: 10px;
    margin: auto;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
import { DataManager, Query } from '@syncfusion/ej2-data';
export default {
  data: function() {
    return {
      data: new DataManager({
        url: 'https://services.syncfusion.com/js/production/api/',
        crossDomain: true
      }),
      query: new
Query().from("ListView").select("EmployeeID,FirstName").take(10),
      fields:  { id: "EmployeeID", text: "FirstName" },
      headerTitle: 'Employees',
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/data-binding/remote-data-cs1" %}

Grouping in Vue Listview component

The ListView supports to wrap the nested element into a group based on the category. The category of each list item can be mapped with groupBy field in the data table, that also supports single-level navigation.

In the following sample, The cars are grouped based on its category by using the groupBy field.

APP.VUE

```
<template>
  <div class="control-section">
     <div id = 'group-list'>
    <!-- Group ListView element -->
    <ejs-listview id='sample-list-group' :dataSource='cars'
:fields='fields'></ejs-listview>
    </div>
  </div>
</template>
<style>
#sample-list-group {
    border: 1px solid #dddddd;
    border-radius: 3px;
    margin: auto;
}
#group-list {
    width: 50%;
    padding: 10px;
    margin: auto;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
export default {
  data: function() {
    return {
      cars: [
    {
        'text': 'Audi A4',
        'id': '9bdb',
        'category': 'Audi'
    },
    {
        'text': 'Audi A5',
        'id': '4589',
        'category': 'Audi'
    },
    {
        'text': 'BMW 501',
        'id': 'f849',
        'category': 'BMW'
    },
    {
        'text': 'BMW 502',
        'id': '7aff',
        'category': 'BMW'
    }
],
fields: { groupBy: 'category', tooltip: 'text' }
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/grouping-cs1" %}

Customization

The grouping header can be customized by using the groupTemplate property for both inline and fixed group header. The complete customization description and explanation with an example is given in the following link.

Group Template.

Check list in Vue Listview component

The ListView supports checkbox in default and group-lists which is used to select multiple items. The checkbox can be enabled by the showCheckBoxproperty.

The Checkbox will be useful in the scenario where we need to select multiple options.For Example, In Shipping cart we can be able to select or unselect the desired items before checkout. It will be useful in selecting multiple items that belongs to same category using the group list.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <div class="control-section">
    <div id = 'flat-list'>
    <!-- ListView element -->
    <ejs-listview id='sample-list-flat' :dataSource='data'
showCheckBox='true' headerTitle='To Do List' showHeader='true'></ejs-
listview>
    </div>
  </div>
</template>
<style>
#sample-list-flat {
    border: 1px solid #dddddd;
    border-radius: 3px;
    margin: auto;
}
#flat-list {
    max-width: 350px;
    padding: 10px;
    margin: auto;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
export default {
  data: function() {
    return {
      data: [
        {text: 'Do Meditation', id: '1'},
        {text: 'Go Jogging', id: '2'},
        {text: 'Buy Groceries', id: '3'},
        {text: 'Pay Phone bill', id: '4'},
        {text: 'Play Football with your friends', id: '5'},
    ],
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/checklist-cs1" %}

Checkbox Position

In ListView the checkbox can be positioned into either Left or Right side of the list-item text. This can be achieved by checkBoxPositon property. By default, checkbox will be positioned to Left of list-item text.

APP.VUE

```
<template>
  <div class="control-section">
    <div id = 'flat-list'>
    <!-- ListView element -->
    <ejs-listview id='sample-list-flat' :dataSource='data'
showCheckBox='true' checkBoxPosition='Right'></ejs-listview>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  </div>
</template>
<style>
#sample-list-flat {
    border: 1px solid #dddddd;
    border-radius: 3px;
    margin: auto;
}
#flat-list {
    width: 50%;
    padding: 10px;
    margin: auto;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
export default {
  data: function() {
    return {
      data: [
        { text: 'Hennessey Venom', id: 'list-01' },
        { text: 'Bugatti Chiron', id: 'list-02' },
        { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
        { text: 'SSC Ultimate Aero', id: 'list-04' },
        { text: 'Koenigsegg CCR', id: 'list-05' },
        { text: 'McLaren F1', id: 'list-06' },
        { text: 'Aston Martin One- 77', id: 'list-07' },
        { text: 'Jaguar XJ220', id: 'list-08' },
        { text: 'McLaren P1', id: 'list-09' },
        { text: 'Ferrari LaFerrari', id: 'list-10' },
      ],
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/checklist-cs2" %}

Nested list in Vue Listview component

The ListView component supports Nested list. For that, the child property should be defined for the nested list in the array of JSON.

APP.VUE

```
<template>
  <div class="control-section">
    <!-- ListView element -->
    <ejs-listview id='listview' :dataSource='continent' :fields='fields'
:headerTitle='headerTitle' showHeader='true'></ejs-listview>
</div>
</template>
<style>
#listview {
    max-width: 400px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
export default {
  data: function() {
    return {
    continent: [
    {
        'text': 'Asia',
        'id': '01',
        'category': 'Continent',
        'child': [{
            'text': 'India',
            'id': '1',
            'category': 'Asia',
            'child': [{
                'text': 'Delhi',
                'id': '1001',
                'category': 'India',
            },
            {
                'text': 'Kashmir',
                'id': '1002',
                'category': 'India',
            },
            {
                'text': 'Goa',
                'id': '1003',
                'category': 'India',
            },
            ]
        },
        {
            'text': 'China',
            'id': '2',
            'category': 'Asia',
            'child': [{
                'text': 'Zhejiang',
                'id': '2001',
                'category': 'China',
            },
            {
                'text': 'Hunan',
                'id': '2002',
                'category': 'China',
            },
            {
                'text': 'Shandong',
                'id': '2003',
                'category': 'China',
            }]
```

```
        }]
    },
    {
        'text': 'North America',
        'id': '02',
        'category': 'Continent',
        'child': [{
            'text': 'USA',
            'id': '3',
            'category': 'North America',
            'child': [{
                'text': 'California',
                'id': '3001',
                'category': 'USA',
            },
            {
                'text': 'New York',
                'id': '3002',
                'category': 'USA',
            },
            {
                'text': 'Florida',
                'id': '3003',
                'category': 'USA',
            }]
        },
        {
            'text': 'Canada',
            'id': '4',
            'category': 'North America',
            'child': [{
                'text': 'Ontario',
                'id': '4001',
                'category': 'Canada',
            },
            {
                'text': 'Alberta',
                'id': '4002',
                'category': 'Canada',
            },
            {
                'text': 'Manitoba',
                'id': '4003',
                'category': 'Canada',
            }]
        }]
    },
    {
        'text': 'Europe',
        'id': '03',
        'category': 'Continent',
        'child': [{
            'text': 'Germany',
            'id': '5',
            'category': 'Europe',
            'child': [{
                'text': 'Berlin',
```

```
                'id': '5001',
                'category': 'Germany',
            },
            {
                'text': 'Bavaria',
                'id': '5002',
                'category': 'Germany',
            },
            {
                'text': 'Hesse',
                'id': '5003',
                'category': 'Germany',
            }]
        }, {
            'text': 'France',
            'id': '6',
            'category': 'Europe',
            'child': [{
                'text': 'Paris',
                'id': '6001',
                'category': 'France',
            },
            {
                'text': 'Lyon',
                'id': '6002',
                'category': 'France',
            },
            {
                'text': 'Marseille',
                'id': '6003',
                'category': 'France',
            }]
        }]
    },
    {
        'text': 'Australia',
        'id': '04',
        'category': 'Continent',
        'child': [{
            'text': 'Australia',
            'id': '7',
            'category': 'Australia',
            'child': [{
                'text': 'Sydney',
                'id': '7001',
                'category': 'Australia',
            },
            {
                'text': 'Melbourne',
                'id': '7002',
                'category': 'Australia',
            },
            {
                'text': 'Brisbane',
                'id': '7003',
                'category': 'Australia',
            }]
```

```
        }, {
            'text': 'New Zealand',
            'id': '8',
            'category': 'Australia',
            'child': [{
                'text': 'Milford Sound',
                'id': '8001',
                'category': 'New Zealand',
            },
            {
                'text': 'Tongariro National Park',
                'id': '8002',
                'category': 'New Zealand',
            },
            {
                'text': 'Fiordland National Park',
                'id': '8003',
                'category': 'New Zealand',
            }]
        }]
    },
    {
        'text': 'Africa',
        'id': '05',
        'category': 'Continent',
        'child': [{
            'text': 'Morocco',
            'id': '9',
            'category': 'Africa',
            'child': [{
                'text': 'Rabat',
                'id': '9001',
                'category': 'Morocco',
            },
            {
                'text': 'Toubkal',
                'id': '9002',
                'category': 'Morocco',
            },
            {
                'text': 'Todgha Gorge',
                'id': '9003',
                'category': 'Morocco',
            }]
        }, {
            'text': 'South Africa',
            'id': '10',
            'category': 'Africa',
            'child': [{
                'text': 'Cape Town',
                'id': '10001',
                'category': 'South Africa',
            },
            {
                'text': 'Pretoria',
                'id': '10002',
                'category': 'South Africa',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            },
            {
                'text': 'Bloemfontein',
                'id': '10003',
                'category': 'South Africa',
            }]
        }]
    }
],
    fields: { tooltip: 'text'},
    headerTitle: 'Continent',
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/nested-list-cs1" %}

Customizing templates in Vue Listview component

The ListView component is designed to customize each list items and group title.

Header template

The ListView header can be customized with the help of headerTemplate property.

To customize header template in your application, set your customized template string to headerTemplate property along with showHeader property as true to display the ListView header.

`ts

let listviewInstance: ListView = new ListView({

data: listData,

headerTemplate: ' Header ',

showHeader: true

})

`

In the below example, we have rendered Listview with customized header which contains search, add and sort buttons.

APP.VUE

```
<template>
  <div class="control-section">
    <div id = 'flat-list'>
    <!-- ListView element -->
    <ejs-listview id='element' :dataSource='data' :headerTemplate='template'
showHeader='true'></ejs-listview>
    </div>
  </div>
</template>
<style>
#element {
    display: block;
```

```
    max-width: 353px;
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
}
#element .e-list-header {
    padding: 0;
}
#element .headerContainer {
    width: 350px;
    height: 48px;
    line-height: 48px;
    background: rgb(2, 120, 215);
    color: white;
    margin-bottom: 3px;
}
.headerContainer .fruitHeader {
    margin-left: 20px;
    font-weight: 500;
    font-size: 22px;
}
.headerContainer #add,
.headerContainer #sort,
.headerContainer #search {
    float: right;
    margin-right: 15px;
    margin-top: 7px;
    background: white;
    color: black
}
.headerContainer .e-search-icon::before {
    content: '\e961';
    color: lightslategray;
}
.headerContainer .e-add-icon::before {
    content: '\e823';
}
.headerContainer .e-sort-icon::before {
    content: '\e840';
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(ListViewPlugin);
Vue.use(ButtonPlugin);
var demoVue = Vue.component("demo", {
  template: `<div class="headerContainer"><span
class="fruitHeader">Fruits</span><ejs-button id="search" iconCss='e-icons e-
search-icon' cssClass='e-small e-round' isPrimary='true'></ejs-button><ejs-
button id="add" iconCss='e-icons e-add-icon' cssClass='e-small e-round'
isPrimary='true'></ejs-button><ejs-button id="sort" iconCss='e-icons e-sort-
icon' cssClass='e-small e-round' isPrimary='true'></ejs-button></div>`,
  data() {
```

```
    return {
      data: {}
    };
  }
});
export default {
  data: function() {
    return {
      data: [
    { text: 'Date', id: '1', imgUrl: './dates.jpg' },
    { text: 'Fig', id: '2', imgUrl: './fig.jpg' },
    { text: 'Apple', id: '3', imgUrl: './apple.png' },
    { text: 'Apricot', id: '4', imgUrl: './apricot.jpg' },
    { text: 'Grape', id: '5', imgUrl: './grape.jpg' },
    { text: 'Strawberry', id: '6', imgUrl: './strawberry.jpg' },
    { text: 'Pineapple', id: '7', imgUrl: './pineapple.jpg' },
    { text: 'Melon', id: '8', imgUrl: './melon.jpg' },
    { text: 'Lemon', id: '9', imgUrl: './lemon.jpg' },
    { text: 'Cherry', id: '10', imgUrl: './cherry.jpg' },
    ],
    template: function () {
                return { template : demoVue}
            }
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/templates/customizing-templates-cs1" %}

Template

The ListView items can be customized with the help of template property.

To customize list items in your application, set your customized template string to template property.

`ts

var demoVue = Vue.component("demo", {

template: `

${text} `,

data() {

return {

data: {}

};

}

});

`

We provided the following built-in CSS classes to customize the list-items. Refer to the following table.

| CSS class | Description |

| ---------- |----------|

| e-list-template, e-list-wrapper | These classes are used to differentiate normal and template rendering, which are mandatory for template rendering. The e-list-template class should be added to the root of the ListView element and e-list-wrapper class should be added to the template element wrapper

 <ejs-listview :cssClass="e-list-template" :template="template"></ejs-listview>
 template: ' '

| e-list-content | This class is used to align list content and it should be added to the content element

 template: ' ' +
 'ListItem' +
 ' '|

| e-list-avatar | This class is used for avatar customization. It should be added to the template element wrapper. After adding it, we can customize our element with Avatar classes

 template: ' ' +
 'MR' +
 'ListItem' +
 ' '|

| e-list-avatar-right | This class is used to align avatar to right side of the list item. It should be added to the template element wrapper. After adding it, we can customize our element with Avatar classes

 template: ' ' +
 'ListItem' +
 'MR' +
 ' '|

| e-list-badge | This class is used for badge customization .It should be added to the template element wrapper. After adding it, we can customize our element with Badge classes

 template: ' '
'ListItem'
'MR'
' '|

| e-list-multi-line | This class is used for multi-line customization. It should be added to the template element wrapper. After adding it, we can customize List item's header and description

template: ' ' +
'ListItem' +
' '|

| e-list-item-header |This class is used to align a list header and it should be added to the header element along with the multi-line class

 template: ' ' +
'ListItem Header' +
'ListItem' +
' '|

In the following example, we have customized list items like Contact app with our avatar component.

APP.VUE

```
{%  raw %}
<template>
<div class="control-section">
<div id = 'flat-list'>
<!-- ListView element -->
<ejs-listview id='List' :dataSource='data' headerTitle='Contacts'
showHeader='true' cssClass='e-list-template' :template='template'
sortOrder='Ascending' width='350px'></ejs-listview>
</div>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<style>
#List {
margin: 0 auto;
border: 1px solid #ccc;
}
#List .e-list-item {
height: 60px;
cursor: pointer;
}
#List .e-list-header .e-text {
font-family: sans-serif;
font-size: 18px;
line-height: 16px;
}
#List #content {
margin: 0;
}
#List .e-list-header{
background: rgb(2, 120, 215);
color: white;
}
#List #info,
#List .name {
font-size: 14px;
margin: 0 60px;
line-height: 20px;
}
#List .name {
padding-top: 8px;
font-weight: 500;
}
.pic01 {
background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/1.png");
}
.pic02 {
background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/3.png");
}
.pic03 {
background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/5.png");
}
.pic04 {
background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/2.png");
}
#List .e-list-item:nth-child(1) .e-avatar {
background-color: #039be5;
}
#List .e-list-item:nth-child(2) .e-avatar {
background-color: #e91e63;
}
#List .e-list-item:nth-child(6) .e-avatar {
background-color: #009688;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
#List .e-list-item:nth-child(8) .e-avatar {
background-color: #0088;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
var demoVue = Vue.component("demo", {
template:
`<div class="e-list-wrapper e-list-multi-line e-list-avatar">
<span :class="['e-avatar e-avatar-circle']" v-if="data.avatar !==
''">{{data.avatar}}</span>
<span :class="[data.pic + ' e-avatar e-avatar-circle']" v-if="data.pic !==
'' "> </span>
<span class="e-list-item-header">{{data.text}}</span>
<span class="e-list-content">{{data.contact}}</span>
</div>`,
data() {
return {
data: {}
};
}
});
export default {
data: function() {
return {
data: [
{
text: "Jenifer",
contact: "(206) 555-985774",
id: "1",
avatar: "",
pic: "pic01"
},
{ text: "Amenda", contact: "(206) 555-3412", id: "2", avatar: "A", pic: ""
},
{
text: "Isabella",
contact: "(206) 555-8122",
id: "4",
avatar: "",
pic: "pic02"
},
{
text: "William ",
contact: "(206) 555-9482",
id: "5",
avatar: "W",
pic: ""
},
{
text: "Jacob",
contact: "(71) 555-4848",
id: "6",
avatar: "",
pic: "pic04"
```

```
},
{ text: "Matthew", contact: "(71) 555-7773", id: "7", avatar: "M", pic: ""
},
{
text: "Oliver",
contact: "(71) 555-5598",
id: "8",
avatar: "",
pic: "pic03"
},
{
text: "Charlotte",
contact: "(206) 555-1189",
id: "9",
avatar: "C",
pic: ""
}
],
fields: {text: 'text'},
template: function () {
return { template : demoVue};
}
};
},
}
</script>
{% endraw %}
```

{% previewsample "page.domainurl/code-snippet/listview/templates/customizing-templates-cs2" %}

Group template

The ListView group header can be customized with the help of groupTemplate property.

To customize the group template in your application, set your customized template string to groupTemplate property.

`ts

var tempVue = Vue.component("demo", {

template: ${items[0].category} ,

data() {

return {

data: {}

};

}

});

`

Copyright © 2001 -2024 Syncfusion Inc.

In the below example, we have grouped Listview based on the category. The category of each list item should be mapped with groupBy field of the data. We have alsodisplayed grouped list items count in the group list header.

APP.VUE

```
{%  raw %}
<template>
<div class="control-section">
<div id = 'flat-list'>
<!-- ListView element -->
<ejs-listview id='List' ref='list' :dataSource='data' :fields='fields'
:template="demoTemplate" :groupTemplate="groupTemplate" cssClass='e-list-
template' width='350px'>
</ejs-listview>
</div>
</div>
</template>
<style>
#List {
display: block;
margin: auto;
border: 1px solid;
border-color: #ccc;
border-color: rgba(0, 0, 0, 0.12);
width: 60%;
}
#List .settings {
height: auto;
}
#List .e-list-item {
height: auto;
padding: 0;
cursor: pointer;
box-sizing: border-box;
}
#List .e-list-header .e-text {
font-family: sans-serif;
font-size: 18px;
line-height: 26px;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
var demoVue = Vue.component("demo", {
template:
`<div class="e-list-wrapper e-list-multi-line e-list-avatar">
<img class="e-avatar e-avatar-circle" :src='data.image'
style="background:#BCBCBC" />
<span class="e-list-item-header">{{data.Name}}</span>
<span class="e-list-content">{{data.contact}}</span>
</div>`,
data() {
return {
data: {}
```

```
};
}
});
var tempVue = Vue.component("demo", {
template:
`<div>
<span class="category">{{data.items[0].category}}</span>
<span class="count"> {{data.items.length}} Item(s)</span>
</div>`,
data() {
return {
data: {}
};
}
});
export default {
data: function() {
return {
data: [
{ Name: 'Nancy', contact:'(206) 555-985774', id: '1', image:
'https://ej2.syncfusion.com/demos/src/grid/images/1.png',  category:
'Experience'},
{ Name: 'Janet', contact: '(206) 555-3412', id: '2', image:
'https://ej2.syncfusion.com/demos/src/grid/images/3.png', category:
'Fresher' },
{ Name: 'Margaret', contact:'(206) 555-8122', id:'4', image:
'https://ej2.syncfusion.com/demos/src/grid/images/4.png', category:
'Experience' },
{ Name: 'Andrew ', contact:'(206) 555-9482', id: '5', image:
'https://ej2.syncfusion.com/demos/src/grid/images/2.png', category:
'Experience'},
{ Name: 'Steven', contact:'(71) 555-4848', id: '6', image:
'https://ej2.syncfusion.com/demos/src/grid/images/5.png', category:
'Fresher' },
{ Name: 'Michael', contact:'(71) 555-7773', id: '7', image:
'https://ej2.syncfusion.com/demos/src/grid/images/6.png', category:
'Experience' },
{ Name: 'Robert', contact:'(71) 555-5598', id: '8', image:
'https://ej2.syncfusion.com/demos/src/grid/images/7.png', category:
'Fresher' },
{ Name: 'Laura', contact:'(206) 555-1189', id: '9', image:
'https://ej2.syncfusion.com/demos/src/grid/images/8.png', category:
'Experience' },
],
fields: {text: 'Name', groupBy: 'category'},
demoTemplate: function () {
return { template : demoVue};
},
groupTemplate: function () {
return { template : tempVue};
},
};
},
}
</script>
{% endraw %}
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/listview/templates/customizing-group-templates-cs1" %}

Virtualization in Vue Listview component

UI virtualization loads only viewable list items in a view port, which will improve the ListView performance while loading a large number of data.

Module injection

In order to use UI Virtualization, `virtualization` module should be injected into the provide section and use `listview` as a key of the object.

```ts
export default {
data: function() {
return {
listData: listData,
enableUi: true
};
},
provide: {
listview: [Virtualization]
}
}
```

Getting started

UI virtualization can be enabled in the ListView by setting the enableVirtualization property to true.

It has two types of scrollers as follows:

Window scroll: This scroller is used in the ListView by default.

Container scroll: This scroller is used, when the height property of the ListView is set.

APP.VUE

```
<template>
  <div class="control-section">
     <ejs-listview id='ui-list' :dataSource='listData'
:enableVirtualization='enableUi' >
    </ejs-listview>
  </div>
</template>
<style>
#app {
    color: #008cff;
    height: 40px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    left: 30%;
    position: absolute;
}
#ui-list {
  display: block;
  max-width: 400px;
  margin: auto;
  border-radius: 3px;
  cursor: pointer;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin, Virtualization } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
let listData = [];
export default {
  beforeCreate: function() {
    listData = [
      { text: "Nancy", id: "0" },
      { text: "Andrew", id: "1" },
      { text: "Janet", id: "2" },
      { text: "Margaret", id: "3" },
      { text: "Steven", id: "4" },
      { text: "Laura", id: "5" },
      { text: "Robert", id: "6" },
      { text: "Michael", id: "7" },
      { text: "Albert", id: "8" },
      { text: "Nolan", id: "9" }
    ];
    for (let i = 10; i <= 1000; i++) {
      let index = parseInt((Math.random() * 10).toString());
      listData.push({ text: listData[index].text, id: i.toString() });
    }
  },
  data: function() {
    return {
      listData: listData,
      enableUi: true
    };
  },
  provide: {
    listview: [Virtualization]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/virtualization/flat-list-cs1" %}

Template

We can use template property to customize list items in UI virtualization.

APP.VUE

```
<template>
  <div class="control-section">
```

```
    <ejs-listview id='ui-list' ref="list" :dataSource='listData'
:showHeader='header' :fields='fields' :headerTitle='title'
        :height='height' :cssClass='cssClass'
:enableVirtualization='enableUi' :template="'gTemplate'">
            <template v-slot:gTemplate="{ data }">
                <div class="e-list-wrapper e-list-avatar" >
                    <span :class="['e-avatar e-avatar-circle ' + data.icon +
' showUI']" v-if="data.imgUrl == undefined">{{ data.icon }}</span>
                    <span :class="['e-avatar e-avatar-circle ' + data.icon +
' hideUI']" v-if="data.imgUrl !== undefined">{{ data.icon }}</span>
                    <img :class="['e-avatar e-avatar-circle showUI']"
:src="data.imgUrl" v-if="data.imgUrl !== undefined" />
                    <img :class="['e-avatar e-avatar-circle hideUI']"
:src="data.imgUrl" v-if="data.imgUrl == undefined" />
                    <span class="e-list-content">{{ data.name }}</span>
                </div>
            </template>
        </ejs-listview>
  </div>
</template>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-lists/styles/material.css";
#app {
    color: #008cff;
    height: 40px;
    left: 30%;
    position: absolute;
}
#ui-list.e-listview {
        margin: auto;
        max-width: 325px;
        line-height: initial;
        border: 1px solid lightgray;
    }
    /* ListView header alignment */
    #ui-list.e-listview .e-list-header {
        height: 50px
    }
    #ui-list.e-listview .e-list-header .e-text {
        line-height: 18px
    }
    /* ListView template customization */
    #ui-list.e-listview .showUI {
        display: flex;
        align-items: center;
        justify-content: center;
        border-radius: 50%;
    }
    #ui-list.e-listview .hideUI {
        display: none;
    }
    #ui-list.e-listview .e-list-item {
        padding: 3px 0;
    }
    #ui-list.e-listview .R {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        background: lightgrey;
    }
    #ui-list.e-listview .M {
        background: pink;
    }
    #ui-list.e-listview .A {
        background: lightgreen;
    }
    #ui-list.e-listview .S {
        background: lightskyblue;
    }
    #ui-list.e-listview .J {
        background: orange;
    }
    #ui-list.e-listview .N {
        background: lightblue;
    }
</style>
<script>
import Vue from "vue";
import { ListViewPlugin, Virtualization } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
let listData = [];
export default {
    beforeCreate: function() {
listData = [
    { name: 'Nancy', icon: 'N', id: '0', },
    { name: 'Andrew', icon: 'A', id: '1' },
    { name: 'Janet', icon: 'J', id: '2' },
    { name: 'Margaret', imgUrl:
'https://ej2.syncfusion.com/react/demos/src/listview/images/margaret.png',
id: '3' },
    { name: 'Steven', icon: 'S', id: '4' },
    { name: 'Laura', imgUrl:
'https://ej2.syncfusion.com/react/demos/src/listview/images/laura.png', id:
'5' },
    { name: 'Robert', icon: 'R', id: '6' },
    { name: 'Michael', icon: 'M', id: '7' },
    { name: 'Albert', imgUrl:
'https://ej2.syncfusion.com/react/demos/src/listview/images/albert.png', id:
'8' },
    { name: 'Nolan', icon: 'N', id: '9' }
];
for (let i = 10; i <= 1000; i++) {
    let index = parseInt((Math.random() * 10).toString());
    listData.push({ name: listData[index].name, icon: listData[index].icon,
imgUrl: listData[index].imgUrl, id: i.toString() });
}
},
  data: function() {
    return {
      listData: listData,
      header: true,
      title: "Contacts",
      fields: { text: "name" },
      height: 500,
      cssClass:'e-list-template',
```

```
      enableUi: true,
    };
  },
  provide: {
    listview: [Virtualization]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/virtualization/template-cs1" %}

Conditional rendering

The following conditional rendering support is provided for the template and groupTemplate.

| Name | Syntax |

| --- | --- | --- |

| conditional class | |

| conditional attribute | |

| conditional text content | ${ $id % 2 === 0 ? 'even-list' : 'odd-list'} |

In the following sample, the light blue is applied for the even list and light coral is applied for the odd list based on the conditional class.

APP.VUE

```
<template>
  <div class="control-section">
     <ejs-listview id='ui-list' :dataSource='listData'
:enableVirtualization='enableUi'  :template='gTemplate' :height='height' >
    </ejs-listview>
  </div>
</template>
<style>
#app {
    color: #008cff;
    height: 40px;
    left: 30%;
    position: absolute;
}
#ui-list {
    display: block;
    max-width: 400px;
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
    cursor: pointer;
}
#list-container{
    height: inherit;
    width: inherit;
    padding-left: 30px;
}
#ui-list .e-list-item{
    padding: 0;
```

```
}
#ui-list .even-list{
  background-color: lightblue;
}
#ui-list .odd-list{
  background-color: lightcoral;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin, Virtualization } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
let listData = [];
export default {
  beforeCreate: function() {
listData = [
    { text: 'Nancy', id: '0', },
    { text: 'Andrew', id: '1' },
    { text: 'Janet', id: '2' },
    { text: 'Margaret', id: '3' },
    { text: 'Steven', id: '4' },
    { text: 'Laura', id: '5' },
    { text: 'Robert', id: '6' },
    { text: 'Michael', id: '7' },
    { text: 'Albert', id: '8' },
    { text: 'Nolan', id: '9' }
];
for (let i = 10; i <= 1000; i++) {
    let index = parseInt((Math.random() * 10).toString());
    listData.push({ text: listData[index].text, id: i.toString() });
}
},
  data: function() {
    return {
      listData: listData,
      gTemplate:  '<div id="list-container" class="${ $id % 2 === 0 ?
\'even-list\' : \'odd-list\' }" > ${text} </div>',
      height: 500,
      enableUi: true,
    };
  },
  provide: {
    listview: [Virtualization]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/virtualization/conditional-ui-cs1" %}

Scrolling in Vue Listview component

Scrolling is a technique that allows you to load more items as the user scrolls through a list, providing a seamless and dynamic user experience.

Copyright © 2001 -2024 Syncfusion Inc.

Render the ListView with dataSource, and bind the scroll event. Within the scroll event, you can access information such as the scroll direction, event name and the distance from the scrollbar to the top and bottom ends through the distanceY parameter.

In the given example, new data is seamlessly added while scrolling. When you scrolls to the bottom and the distance scrolled is less than 100 pixels, it dynamically loads a batch of items into the list as long as there are more items to render.

APP.VUE

```
<template>
  <div class="control-section">
    <div class="grid-container"><div><h3>Chat</h3>
    <!-- ListView element -->
    <ejs-listview id='List' ref='list' :dataSource='visibleData'
cssClass='e-list-template' :template='template' width='400px' height='320px'
:scroll="onListScrolled"></ejs-listview>
    </div>
    </div>
  </div>
</template>
<style>
.grid-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-gap: 20px;
    align-items: center;
  }

  h3 {
    margin: 0;
  }

  /* Optional: Add styling to headings and divs */
  .grid-container > div {
    border: 1px solid #ddd;
    padding: 20px;
    border-radius: 5px;
    background-color: #f4f4f4;
  }
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-lists/styles/material.css";
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
var demoVue = Vue.component("App", {
  template:
    `<div :style="{ display: 'flex', justifyContent:
getJustifyContentStyle(data.positionClass) }" class="e-list-wrapper e-list-
multi-line">
        <span style="display: block; white-space: normal; max-width: 80%;
padding: 10px; background-color: #e0e0e0; border-radius: 10px; word-wrap:
break-word;" class="e-list-item-header">
          {{ data.text }}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </span>
      </div>`,
  data() {
    return {
      data: {}
    };
  },
  methods:{
    getJustifyContentStyle(positionClass) {
      return positionClass === "right" ? "flex-end" : "";
    },
  }
});
export default {
  data: function() {
    return {
      data: [
  {
    text: "Hi Guys, Good morning! \uD83D\uDE0A, I'm very delighted to share
with you the news that our team is going to launch a new mobile
application",
    positionClass: 'right',
  },
  {
    text: "Oh! That's great \uD83D\uDE42",
    positionClass: 'left',
  },
  {
    text: 'That is a good news \uD83D\uDE00',
    positionClass: 'right',
  },
  {
    text: 'What kind of application we are gonna launch? \uD83E\uDD14',
    positionClass: 'left',
  },
  {
    text: 'A kind of "Emergency Broadcast App" like being able to just
invite someone to teams without it impacting how many people have official
access',
    positionClass: 'right',
  },
  {
    text: 'Who will be the client users for our applications? ',
    positionClass: 'left',
  },
  {
    text: 'Is currently the only way to invite someone through 0365? Just
wondering down the road how organization would want to handle that with
freelancers, like being able to just invite someone to teams without it
impacting how many people have official access \uD83D\uDE14',
    positionClass: 'right',
  },
  {
    text: 'Yes, however, that feature of inviting someone from outside your
organization is planned - expected closer to GA next year \uD83D\uDC4D',
    positionClass: 'left',
  },
```

```
  {
    text: 'I guess we should switch things over to hear for a while. How
long does the trial last? \uD83E\uDD14',
    positionClass: 'right',
  },
  { text: 'I think the trial is 30 days. \uD83D\uDE03', positionClass:
'left' },
  {
    text: 'Only 0365 only members of your organization. They said that they
are listening to customer feedback and hinted that guest users would be
brought in down the road \uD83D\uDE09',
    positionClass: 'right',
  },
  { text: 'Cool thanks! \uD83D\uDC4C', positionClass: 'left' },
],
      visibleData: [],
      remainingData: [],
  template: function () {
              return { template : demoVue};
          }
  };
  },
  watch: {
    data: {
      handler(newVal) {
        this.visibleData = newVal.slice(0, 5);
        this.remainingData = newVal.slice(5);
      },
      immediate: true,
    },
  },
  methods:{
     onListScrolled: function (args) {
      if (args.scrollDirection === "Bottom" && args.distanceY < 100) {
        const additionalItems = this.remainingData.slice(0, 5);
        this.visibleData = this.visibleData.concat(additionalItems);
        this.remainingData = this.remainingData.slice(5);
      }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/scrolling" %}

Accessibility in Vue Listview component

The ListView component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the ListView component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

Copyright © 2001 -2024 Syncfusion Inc.

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The ListView component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the ListView component:

| Attributes | Purpose |

| --- | --- |

| `role=list` | Specifies the non selectable list container. |

| `role=listitem`| Specifies the role of each item in a selectable ListView and its containment within the list. |

| role=presentation | Specifies the role of non selectable list element. |

| role=checkbox | Indicates checkbox control along with listitem element. |

| aria-checked | Indicates the current checked state of checkbox. |

| aria-label | Provides an accessible name for the ListView Checkbox. |

| aria-disabled | Indicates element is perceivable but disabled. |

Keyboard interaction

The ListView component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the ListView component.

| Keyboard shortcuts | Actions |

|------------|-------------------|

| Arrow Up | Move to the previous list item |

| Arrow Down | Move to the next list item |

| Space | Check and uncheck the targeted list from the whole list |

| BackSpace | Get back to the previous lists if it is nested list |

| Home | Moves focus to first list item |

| End | Moves focus to last list item |

Ensuring accessibility

The ListView component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the ListView component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the ListView component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/list-view.html" %}

See also

- Accessibility in Syncfusion Vue components

Style in Vue Listview component

The following content provides the exact CSS structure that can be used to modify the control’s appearance based on user preference.

Customizing ListView

Use the following CSS to customize the ListView.

`css

.e-listview {

border: 5px solid rgb(173, 255, 47);

}

Copyright © 2001 -2024 Syncfusion Inc.

`

Customizing the list items

Use the following CSS to customize the items of ListView.

`css

.e-listview .e-list-item {

text-align: center;

color: pink;

background-color: #2fa1ff;

}

`

Customizing ListView's header

Use the following CSS to customize the header of ListView control.

`css

.e-listview .e-list-header{

color: #2fa1ff;

justify-content: center;

}

`

Customizing group header of ListView

Use the following CSS to customize the category of the group items.

`css

.e-listview .e-list-group-item {

color: rgb(173, 255, 47);

background-color: maroon;

text-align: end;

}

`

Customizing the hover state of ListView control

Use the following CSS to customize the list item when hovering.

Customizing ListView hover state with the checkbox checked

`css

.e-listview .e-list-item.e-hover.e-active.e-checklist {

color: rgb(83, 5, 79);

background-color: rgb(173, 255, 47);

Copyright © 2001 -2024 Syncfusion Inc.

}
`

Customizing ListView hover state
`css
.e-listview .e-list-item.e-hover {
color:red;
background-color: rgb(173, 255, 47);
}
`

Customizing selected item of ListView control

Use the following CSS to customize the selected list item.

Customizing ListView's selected item with the checkbox checked
`css
.e-listview .e-list-item.e-checklist.e-focused.e-active {
color: rgb(83, 5, 79);
background-color: rgb(0, 15, 100);
}
`

Customizing ListView's selected item
`css
.e-listview .e-list-item.e-focused {
color: #2fa1ff;
background-color: rgb(0, 15, 100);
}
`

How To

Get selected items from listview in Vue Listview component

Single or many items can be selected by users in the ListView component. An API is used to get selected items from the list items. This is called as the getSelectedItems method.

getSelectedItems method

This is used to get the details of the currently selected item from the list items. It returns the SelectedItem | SelectedCollection

The getSelectedItems method returns the following items from the selected list items.

Return type	Purpose

| text | Returns the text of selected item lists |

| data | Returns the whole data of selected list items, i.e., returns the fields data of selected li.|

| item | Returns the collections of list items |

APP.VUE

```
<template>
  <div class="control-section">
    <ejs-listview ref='listview' id='sample-list' :dataSource='data'
showCheckBox=true :fields='fields'></ejs-listview>
    <br/>
    <ejs-button id="btn"  v-on:click.native="onClick" >Get Selected
Items</ejs-button>
    <div ref='valEle' id="val"> </div>
  </div>
</template>
<style>
#sample-list {
  display: block;
  max-width: 400px;
  margin: auto;
  border: 1px solid #dddddd;
  border-radius: 3px;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ListViewPlugin);
Vue.use(ButtonPlugin);
var demoVue = Vue.component("demo", {
  template: `<div class='text-content'> {{data.text}} <span class = 'delete-
icon'></span> </div>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data: function() {
    return {
      data: [
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02', isChecked: true },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03'},
    { text: 'SSC Ultimate Aero', id: 'list-04', isChecked: true  },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07', isChecked: true },
    { text: 'Jaguar XJ220', id: 'list-08' }],
    fields: { id: 'id', isChecked:'isChecked' },
    };
  },
  methods: {
```

```
    onClick: function(event){
      let selecteditem =this.$refs.listview.getSelectedItems();
      this.$refs.valEle.innerHTML="";
      for(let i=0; i< selecteditem["data"].length; i++) {
      let listData = document.createElement('p');
      listData.innerHTML = "text : "+ selecteditem["text"][i]+" , "+"id :
"+selecteditem["data"][i].id;
      this.$refs.valEle.append(listData);
      }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/checklist-cs5" %}

Use dynamic templates in listview based on device in Vue Listview component

The Syncfusion Essential JS2 components are desktop and mobile-friendly. So, you can use Syncfusion components in both modes. The component templates are not always fixed. Applications may need to load various templates depending upon the device.

Integration

In the ListView component, template support is being used. In some cases, the component wrapper is always responsive across all devices, but the template contents are dynamically changed with unspecified (sample side) dimensions. CSS customization is also needed in sample-side to align template content responsively in both mobile and desktop modes. Here, two templates have been loaded for mobile and desktop modes. To check the device mode, a browser module has been imported from the ej2-base package.

APP.VUE

```
<template>
  <div class="control-section">
    <div id = 'flat-list'>
    <!-- ListView element -->
    <ejs-listview id='List' ref='list' :dataSource='data' :fields='fields'
:template="templatecheck ? mobtemplate : wintemplate"
headerTitle='Syncfusion Blog' showHeader='true'>
    </ejs-listview>
    </div>
  </div>
</template>
<style>
@import './template.css';
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import { Browser } from '@syncfusion/ej2-base';
Vue.use(ListViewPlugin);
var demoVue = Vue.component("demo", {
  template: `<div class="settings">
        <div id="postContainer">
            <div id="postImg">
                <img :src='data.image' /></div>
```

```
            <div id="content">
                <div id="info">
                    <div id="logo">
                        <div id="share">
                            <span class="share"></span> </div>
                        <div id="comments"> <span class="comments"></span>
</div>
                        <div id="bookmark"> <span class="bookmark"></span>
</div>
                    </div>
                </div>
                <div class="name">{{data.Name}}</div>
                <div class="description">{{data.content}}</div>
                <div class="timeStamp">{{data.timeStamp}} </div>
            </div>
        </div>
    </div>`,
  data() {
    return {
      data: {}
    };
  }
});
var tempVue = Vue.component("demo", {
  template: `<div class="settings">
        <div id="postContainer">
            <div id="postImg">
                <img :src='data.image' /></div>
            <div id="content">
                <div class="name">{{data.Name}}</div>
                <div class="description">{{data.content}}</div>
                <div id="info">
                    <div id="logo">
                        <div id="share">
                            <span class="share"></span> </div>
                        <div id="comments"> <span class="comments"></span>
</div>
                        <div id="bookmark"> <span class="bookmark"></span>
</div>
                    </div>
                    <div class="timeStamp">{{data.timeStamp}} </div>
                </div>
            </div>
        </div>
    </div>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data: function() {
    return {
    data: [
        { Name: 'IBM Open-Sources Web Sphere Liberty Code', content: 'In
September, IBM announced that it would be open-sourcing the code for
```

```
WebSphere...', id: '1', image:
'https://ej2.syncfusion.com/demos/src/listview/images/1.png', timeStamp:
'Syncfusion Blog - October 19, 2017' },
        { Name: 'Must Reads: 5 Big Data E-books to upend your development',
content: 'Our first e-book was published in May 2012-jQuery Succinctly was
the start of over...', id: '2', image:
'https://ej2.syncfusion.com/demos/src/listview/images/2.png', timeStamp:
'Syncfusion Blog - October 18, 2017'  },
        { Name: 'The Syncfusion Global License: Your Questions, Answered ',
content: 'Syncfusion recently hosted a webinar to cover the ins and outs of
the Syncfusion global...', id: '4', image:
'https://ej2.syncfusion.com/demos/src/listview/images/3.png', timeStamp:
'Syncfusion Blog - October 18, 2017'  },
        { Name: 'Know: What is Coming from Microsoft this Fall ', content:
'On October 17, Microsoft will release its Fall Creators Update for the
Windows 10 platform...', id: '5', image:
'https://ej2.syncfusion.com/demos/src/listview/images/6.png', timeStamp:
'Syncfusion Blog - October 17, 2017'  }],
    fields: { text:'Name' },
    mobtemplate: function () {
      return { template : demoVue}
    },
    wintemplate: function () {
      return { template : tempVue}
    },
    templatecheck: Browser.isDevice,
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/dynamic-template-cs1" %}

Create dual list from listview in Vue Listview component

The dual list contains two ListView. This allows you to move list items from one list to another using the client-side

events. This section explains how to integrate the ListView component to achieve dual list.

Use cases

- Stock exchanges of two different countries
- Job applications (skill sets)

Integration of Dual List

Here, two ListView components have been used to display the list items. An ej2-button is used to transfer data between the ListView, and a textbox is used to achieve the UI of filtering support.

The dual list supports:

- Moving whole data from one list to another.
- Moving selected data from one list to another.
- Filtering the list by using a client-side typed character.

Copyright © 2001 -2024 Syncfusion Inc.

In the ListView component, sorting is enabled using the sortOrder property, and the select event is triggered while selecting an item. Here, the select event is triggered to enable and disable button states.

Manipulating data

Moving whole data from the first list to the second list(>>)

- Here, the whole data can be moved from the first ListView to the second by clicking the first button. When clicking the button, the whole list items are sliced, and concat with the second ListView. This button is enabled only when the data source of the first ListView is not empty.

Moving whole data from the second list to the first list(<<)

- The functionality of the second button is the same as above, and data is transferred from the second list to the first list. This button is enabled only when the data source of the second ListView is not empty.

Moving selected item from one list to another list (>) and (<)

- The Select event is triggered when selecting a list item in the ListView. The selected items can be transferred between two lists. These buttons will be enabled when selecting an item in lists.

Filtering method

- The filtering method is used to filter list items when typing a character in the text box. In this method, the dataManager has been used to fetch data from the data source and display in ListView.

Sorting

- By using the dual list, list items can be sorted in the ListView component using the sortOrder property. You can enable sorting in one ListView; in the same order, data can be transferred to another ListView.

APP.VUE

```
<template>
 <div>
            <div id="text1">
            <input ref='textboxEle' class="e-input" type="text"
id="firstInput" placeholder="Filter" title="Type in a name"
@keyup="onFirstKeyUp" />
              </div>
            <ejs-listview ref='firstListObj' id='list-1'
:dataSource='firstListData' :fields='fields' sortOrder='Ascending'
:select="onFirstListSelect"></ejs-listview>
             <div id="btn">
             <ejs-button ref='firstBtnObj' id="firstBtn"
@click.native="firstbtnclick"> >> </ejs-button>
             <ejs-button ref='secondBtnObj' id="secondBtn"
:disabled='disabled' @click.native="secondbtnclick"> > </ejs-button>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <ejs-button ref='thirdBtnObj' id="thirdBtn"
:disabled='disabled' @click.native="thirdbtnclick"> < </ejs-button>
            <ejs-button ref='fourthBtnObj' id="fourthBtn"
@click.native="fourthbtnclick"> << </ejs-button>
            </div>
            <div id="text2">
            <input ref='textEle' class="e-input" type="text"
id="secondInput" placeholder="Filter" title="Type in a name"
@keyup="onSecondKeyUp" />
            </div>
            <ejs-listview ref='secondListObj' id='list-2'
:dataSource='secondListData' :fields='fields' sortOrder='Ascending'
:select="onSecondListSelect"></ejs-listview>
</div>
</template>
<style>
#list-1,
    #list-2 {
        width: 40%;
        height: 430px;
        box-shadow: 0 1px 4px #ddd;
        border-bottom: 1px solid #ddd;
    }
    #firstList, #secondList {
        margin-top: 13px;
    }
    .e-btn {
        width: 60px;
        height: 60px;
        margin-bottom: 15px;
    }
    #btn {
        float: left;
        width: 5%;
        padding-left: 30px;
        margin-top: 67px;
    }
    #list-1 {
        float: left;
    }
    #list-2 {
        float: right;
    }
    #firstInput {
        width: 40%;
    }
    #secondInput {
      margin-left: 102px;
      width: 51.4%;
    }
    #text2 {
        margin-top: -23px;
        margin-left: 194px;
    }
</style>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
import { DataManager, Query, ODataV4Adaptor } from "@syncfusion/ej2-data";
enableRipple(true);
Vue.use(ListViewPlugin);
Vue.use(ButtonPlugin);
Vue.use(ListViewPlugin);
export default {
  data: function() {
      return {
      firstListData: [
  { text: "Hennessey Venom", id: "list-01" },
  { text: "Bugatti Chiron", id: "list-02" },
  { text: "Bugatti Veyron Super Sport", id: "list-03" },
  { text: "SSC Ultimate Aero", id: "list-04" },
  { text: "Koenigsegg CCR", id: "list-05" },
  { text: "McLaren F1", id: "list-06" }
],
secondListData : [
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
  ],
      fields:  {text: "text", id: "id" },
      disabled: true
    };
  },
  mounted: function(){
      this.firstListData = this.$refs.firstListObj.dataSource.slice();
      this.secondListData = this.$refs.secondListObj.dataSource.slice();
  },
  methods: {
    firstbtnclick: function() {
        this.$refs.secondListObj.dataSource =
Array.prototype.concat.call(this.$refs.firstListObj.dataSource,
this.$refs.secondListObj.dataSource);
        this.$refs.secondListObj.dataBind();
        this.updateFirstListData();

this.$refs.firstListObj.removeMultipleItems(this.$refs.firstListObj.$el.ej2_
instances[0].liCollection);
        this.firstListData =
this.firstListData.concat(this.$refs.firstListObj.dataSource);
        this.secondListData = this.$refs.secondListObj.dataSource.slice();
        this.$refs.firstBtnObj.$el.disabled = true;
        this.onFirstKeyUp();
        this.setButtonState();
    },
  secondbtnclick:function() {   //Here, the selected list items are moved to
the second list on clicking move button
        let e = this.$refs.firstListObj.getSelectedItems();
        this.$refs.secondListObj.dataSource =
Array.prototype.concat.call(this.$refs.secondListObj.dataSource, e.data);
        this.$refs.firstListObj.removeItem(e.item);
        this.firstListData = this.$refs.firstListObj.dataSource;
```

```
        this.secondListData = this.$refs.secondListObj.dataSource.slice();
        this.onFirstKeyUp();
        this.$refs.secondBtnObj.$el.disabled = true;
        this.setButtonState();
    },
   thirdbtnclick: function() {  //Here, the selected list items are moved to
the first list on clicking move button
        let e = this.$refs.secondListObj.getSelectedItems();
        this.$refs.firstListObj.dataSource =
Array.prototype.concat.call(this.$refs.firstListObj.dataSource, e.data);
        this.$refs.secondListObj.removeItem(e.item);
        this.secondListData = this.$refs.secondListObj.dataSource;
        this.firstListData = this.$refs.firstListObj.dataSource.slice();
        this.onSecondKeyUp();
        this.$refs.thirdBtnObj.$el.disabled = true;
        this.setButtonState();
    },
   fourthbtnclick: function() {   //Here, all list items are moved to the
first list on clicking move all button
        this.$refs.firstListObj.dataSource =
Array.prototype.concat.call(this.$refs.firstListObj.dataSource,
this.$refs.secondListObj.dataSource);
        this.updateSecondListData();

this.$refs.secondListObj.removeMultipleItems(this.$refs.secondListObj.$el.ej
2_instances[0].liCollection);
        this.secondListData =
this.secondListData.concat(this.$refs.secondListObj.dataSource);
        this.firstListData = this.$refs.firstListObj.dataSource.slice();
        this.onSecondKeyUp();
        this.setButtonState();
    },
    updateFirstListData: function() {  //Here, the ListView data source is
updated to the first list

Array.prototype.forEach.call(this.$refs.firstListObj.$el.ej2_instances[0].li
Collection, (list) => {
            this.firstListData.forEach((data, index) => {
                if (list.innerText.trim() === data.text) {
                  this.firstListData.splice(index, 1)
                }
            });
        });
        this.$refs.textboxEle.value= '';
        let ds = [];
        this.firstListData.forEach((data) => {
            ds.push(data);
        });
        this.firstListData = ds;
    },
    //Here, the ListView dataSource is updated for the second list
    updateSecondListData: function() {

Array.prototype.forEach.call(this.$refs.secondListObj.$el.ej2_instances[0].l
iCollection, (list) => {
            this.secondListData.forEach((data, index) => {
                if (list.innerText.trim() === data.text){
```

```
                    this.secondListData.splice(index, 1);
                }
            });
        });
        this.$refs.textEle.value = '';
        let ds = [];
        this.secondListData.forEach((data) => {
            ds.push(data);
        });
        this.secondListData = ds;
    },
    onFirstListSelect: function() {
        this.$refs.secondBtnObj.disabled = false;
    },
    onSecondListSelect: function() {
        this.$refs.thirdBtnObj.disabled = false;
    },
    //Here, filtering is handled using the dataManager for the first list
    onFirstKeyUp: function(e) {
        let value = this.$refs.textboxEle.value;
        let data = new DataManager(this.firstListData).executeLocal(new
Query().where('text', 'startswith', value, true));
        if (!value) {
            this.$refs.firstListObj.dataSource = this.firstListData.slice();
        } else {
            this.$refs.firstListObj.dataSource = data;
        }
    },
    //Here, filtering is handled using the dataManager for the second list
     onSecondKeyUp:function(e) {
        let value =this.$refs.textEle.value;
        let data = new DataManager(this.secondListData).executeLocal(new
Query().where('text', 'startswith', value, true));
        if (!value) {
            this.$refs.secondListObj.dataSource =
this.secondListData.slice();
        } else {
            this.$refs.secondListObj.dataSource = data;
        }
    },
    //Here, the state of the button is changed
    setButtonState: function() {
        if (this.$refs.firstListObj.dataSource.length) {
            this.$refs.firstBtnObj.$el.disabled = false;
        } else {
            this.$refs.firstBtnObj.$el.disabled = true;
            this.$refs.secondBtnObj.$el.disabled = true;
        }
        if (this.$refs.secondListObj.dataSource.length) {
            this.$refs.fourthBtnObj.$el.disabled = false;
        } else {
            this.$refs.fourthBtnObj.$el.disabled = true;
            this.$refs.thirdBtnObj.$el.disabled = true;
        }
    }
  }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/checklist-cs3" %}

Load list items in child list dynamically in Vue Listview component

To load list items in child list dynamically, push the new list item data into the existing dataSource using the select event.

Refer to the following steps to load list item into the child list:

1. Initially, render the ListView with the required data source.
2. Bind the select event that triggers selecting list item in the ListView component. By using the select event, you can push the new list item to the child list of the data source on specifying its item index. Item index can be obtained from the SelectEventArgs of the select event.

APP.VUE

```
<template>
  <div class="control-section">
    <div id = 'flat-list'>
    <!-- ListView element -->
    <ejs-listview id='listview' ref='list' :dataSource='data'
:fields='fields' headerTitle='Folders' showHeader='true' showIcon='true'
:select='onSelect'>
    </ejs-listview>
    </div>
  </div>
</template>
<style>
#listview {
    display: block;
    max-width: 400px;
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
}
#listview.e-listview .e-list-icon {
    height: 24px;
    width: 30px;
}
.folder, .file  {
    background:
url('https://ej2.syncfusion.com/demos/src/listview/images/file_icons.png')
no-repeat;
    background-size: 300%;
}
.folder{
    background-position: -5px -461px;
}
.file {
    background-position: -5px -151px;
}
.list {
    color:deeppink !important;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
export default {
  data: function() {
    return {
      data:[
    {
        id: '01', text: 'Music', icon: 'folder',
        child: [
            { id: '01-01', text: 'Gouttes.mp3', icon: 'file' }
        ]
    },
    {
        id: '02', text: 'Videos', icon: 'folder',
        child: [
            { id: '02-01', text: 'Naturals.mp4', icon: 'file' },
            { id: '02-02', text: 'Wild.mpeg', icon: 'file' },
        ]
    },
    {
        id: '03', text: 'Documents', icon: 'folder',
        child: [
            { id: '03-01', text: 'Environment Pollution.docx', icon: 'file'
},
            { id: '03-02', text: 'Global Water, Sanitation, & Hygiene.docx',
icon: 'file' },
            { id: '03-03', text: 'Global Warming.ppt', icon: 'file' },
            { id: '03-04', text: 'Social Network.pdf', icon: 'file' },
            { id: '03-05', text: 'Youth Empowerment.pdf', icon: 'file' },
        ]
    },
    {
        id: '04', text: 'Pictures', icon: 'folder',
        child: [
            {
                id: '04-01', text: 'Camera Roll', icon: 'folder',
                child: [
                    { id: '04-01-01', text: 'WIN_20160726_094117.JPG', icon:
'file' },
                    { id: '04-01-02', text: 'WIN_20160726_094118.JPG', icon:
'file' },
                    { id: '04-01-03', text: 'WIN_20160726_094119.JPG', icon:
'file' }
                ]
            },
            {
                id: '04-02', text: 'Wind.jpg', icon: 'file'
            },
            {
                id: '04-02', text: 'Stone.jpg', icon: 'file'
            },
            {
                id: '04-02', text: 'Home.jpg', icon: 'file'
            },
```

```
            {
                id: '04-02', text: 'Bridge.png', icon: 'file'
            }
        ]
    },
    {
        id: '05', text: 'Downloads', icon: 'folder',
        child: [
            { id: '05-01', text: 'UI-Guide.pdf', icon: 'file' },
            { id: '05-02', text: 'Tutorials.zip', icon: 'file' },
            { id: '05-03', text: 'Game.exe', icon: 'file' },
            { id: '05-04', text: 'TypeScript.7z', icon: 'file' },
        ]
    },
],
    fields: { iconCss: 'icon', tooltip: 'text'},
    };
  },
  methods:{
    onSelect: function(args){
      this.$refs.list.dataSource[args.index].child.push({ id: '01-02', text:
'Newly Added File', icon: 'file', htmlAttributes: { role: 'li', class:
'list' } });
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/checklist-cs7" %}

Customize listview with dynamic tags in Vue Listview component

You can customize the ListView items using the template property. Here, the dynamic tags are added and removed in the list item from another ListView. Refer to the following steps to achieve this.

*Initialize dynamic ListView with required property that holds the tags of parent ListView, and bind the select event (triggers when the list item is selected), in which you can get and add the selected item value as tags into parent ListView. Refer to the following code sample.

`ts

//Select the event that is is rendered inside dialog for ListView

addTag: function(e) {

let listTag = document.createElement('span');

listTag.className = 'advanced-option';

let labelElem = document.createElement('span');

labelElem.className = 'label';

let deleteElem = document.createElement('span');

deleteElem.className = 'delete';

deleteElem.onclick = this.removeTag;

labelElem.innerHTML = e.target.textContent;

Copyright © 2001 -2024 Syncfusion Inc.

```
listTag.appendChild(labelElem);
listTag.appendChild(deleteElem);
let tag = document.createElement('span');
tag.className = 'advanced-option-list';
tag.appendChild(listTag);
this.$refs.listviewInstance.$el.ej2_instances[0].element.querySelector('.e-active').appendChild(tag);
}
`
```

- Render the dialog component with empty content and append the created dynamic ListView object to the dialog on created event.
- Bind the click event for button icon (+) to update the ListView data source with tags, and open the dialog with this dynamic ListView. Refer to the following code sample.

```ts
//Method to hide/show the dialog and update the ListView data source
renderDialog: function(id){
if (document.getElementsByClassName('e-popup-open').length !== 0) {
this.$refs.dialogInstance.hide();
}
else {
this.listObj = document.getElementById('dialog').querySelector("#list").ej2_instances[0];
this.listObj.dataSource = this.datasource[id];
this.listObj.dataBind();
this.$refs.dialogInstance.position = { X: document.querySelector('.e-add-icon').getBoundingClientRect().left + 50, Y: document.querySelector('.e-add-icon').getBoundingClientRect().top - 5 };
this.$refs.dialogInstance.show();
}
}
`
```

- Bind the click event with added dynamic tags to remove it. Refer to the following code sample.

```ts
//Method to remove the list item
removeTag() {
```

this.$refs.listviewInstance.$el.ej2_instances[0].selectedLI.children[1].remove();

}

`

APP.VUE

```
<template>
  <div id="sample">
    <ejs-listview ref='listviewInstance' id='templatelist'
:dataSource='data' :fields='fields' width='350' :template='listTemplate'>
    </ejs-listview>
     <ejs-dialog id='dialog' ref='dialogInstance' width='200px'
:content='dialogcontent' :animationSettings='animation' :visible='visible'
showCloseIcon='true' :position='position' :created='createList'>
      </ejs-dialog>
  </div>
</template>
<style>
#sample{
    padding: 40px;
}
.advanced-option-list {
    display: inline-flex;
    list-style: none;
    margin: 0;
    padding: 0;
    margin-top: 4px;
    overflow: auto;
}
.advanced-option {
    border: 1px solid #0078d7;
    -webkit-border-radius: 2px;
    -moz-border-radius: 2px;
    border-radius: 2px;
    padding: 0 20px 0 5px;
    margin: 0 3px 1px 0;
    float: left;
    position: relative;
    background: white;
    line-height: 21px;
}
.advanced-option .label {
    color: #0078d7;
    font-size: 10px;
}
.advanced-option .delete {
    background: url(https://research.isg-
one.com/Assets/img/Content/icn/tiny-del.png) right center no-repeat;
    padding-left: 10px;
    padding-top: 2px;
    right: 0;
    position: absolute;
    top: 1px;
    cursor: pointer;
    height: 16px;
    width: 30px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
.designationstyle {
    float: right;
    position: relative;
    right: 10px;
}
.cont-bg {
    font-size: 17px;
    height: 46px;
    padding-top: 10px;
    width: 100%;
}
.designationstyle .e-add-icon::before {
    content: '\e823';
}
#sample .e-list-item {
    height: auto;
}
.e-dialog .e-dlg-header-content+.e-dlg-content {
    margin-top: -18px;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { DialogPlugin } from "@syncfusion/ej2-vue-popups";
Vue.use(ListViewPlugin);
Vue.use(ButtonPlugin);
Vue.use(DialogPlugin);
var listVue = Vue.component("demo", {
  template: ` <div><span class="templatetext">{{data.Name}} </span> <span
class="designationstyle"><ejs-button :id='data.Id' class="e-but" iconCss='e-
icons e-add-icon' cssClass='e-small e-round'></ejs-button></span></div>`,
  data() {
    return {
      data: {}
    };
  }
});
var dialogVue = Vue.component("demo", {
  template: `<ejs-listview id="list" showHeader=true headerTitle='Favorite'
width='200px'></ejs-listview>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data: function() {
         return {
    data: [{ "Id": "Brooke", "Name": "Brooke" },
{ "Id": "Claire", "Name": "Claire" },
{ "Id": "Erik", "Name": "Erik" },
{ "Id": "Grace", "Name": "Grace" },
{ "Id": "Jacob", "Name": "Jacob" }],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    fields: { text:'Name' },
    listTemplate: function () {
      return { template : listVue}
    },
    brookeTag : [{ "id": "list11", "Name": "Discover Music" },
{ "id": "list12", "Name": "Sales and Events" },
{ "id": "list13", "Name": "Categories" },
{ "id": "list14", "Name": "MP3 Albums" },
{ "id": "list15", "Name": "More in Music" },
],
    claireTag : [{ "id": "list21", "Name": "Songs" },
{ "id": "list22", "Name": "Bestselling Albums" },
{ "id": "list23", "Name": "New Releases" },
{ "id": "list24", "Name": "Bestselling Songs" },
],
    erikTag : [{ "id": "list31", "Name": "Artwork" },
{ "id": "list32", "Name": "Abstract" },
{ "id": "list33", "Name": "Acrylic Mediums" },
{ "id": "list34", "Name": "Creative Acrylic" },
{ "id": "list35", "Name": "Canvas Art" }
],
    graceTag : [{ "id": "list41", "Name": "Rock" },
{ "id": "list42", "Name": "Gospel" },
{ "id": "list43", "Name": "Latin Music" },
{ "id": "list44", "Name": "Jazz" },
],
    jacobTag: [{ "id": "list51", "Name": "100 Albums - $5 Each" },
{ "id": "list52", "Name": "Hip-Hop and R&B Sale" },
{ "id": "list53", "Name": "CD Deals" }
],
    visible: false,
    datasource: { },
    animation: {effect: 'None'},
    dialogcontent: function () {
      return { template : dialogVue}
    },
    position: {},
    };
  },
  created(){
    this.datasource = { "Brooke": this.brookeTag, "Claire": this.claireTag,
"Erik": this.erikTag, "Grace": this.graceTag, "Jacob": this.jacobTag };
  },
  mounted(){
      for (let i = 0; i < this.data.length; i++) {
        document.getElementById(this.data[i].Id).addEventListener("click",
(e) =>{
        this.renderDialog(e.currentTarget.id);
    });
}
  },
  methods: {
    createList: function(e) {
    let listElem = document.getElementById('dialog').querySelector("#list");
    let listObj =
document.getElementById('dialog').querySelector("#list").ej2_instances[0];
    listObj.dataSource = this.datasource.Brooke;
```

```
    listObj.fields = this.fields;
    listObj.addEventListener('select', ()=> { this.addTag(event);});
    listObj.appendTo(listElem);
    },
    renderDialog: function(id){
     if (document.getElementsByClassName('e-popup-open').length !== 0) {
        this.$refs.dialogInstance.hide();
    }
    else {
        this.listObj =
document.getElementById('dialog').querySelector("#list").ej2_instances[0];
        this.listObj.dataSource = this.datasource[id];
        this.listObj.dataBind();
        this.$refs.dialogInstance.position = { X:
document.querySelector('.e-add-icon').getBoundingClientRect().left + 50, Y:
document.querySelector('.e-add-icon').getBoundingClientRect().top - 5 };
        this.$refs.dialogInstance.show();
    }
   },
   addTag: function(e) {
    let listTag = document.createElement('span');
    listTag.className = 'advanced-option';
    let labelElem = document.createElement('span');
    labelElem.className = 'label';
    let deleteElem = document.createElement('span');
    deleteElem.className = 'delete';
    deleteElem.onclick = this.removeTag;
    labelElem.innerHTML = e.target.textContent;
    listTag.appendChild(labelElem);
    listTag.appendChild(deleteElem);
    let tag = document.createElement('span');
    tag.className = 'advanced-option-list';
    tag.appendChild(listTag);

this.$refs.listviewInstance.$el.ej2_instances[0].element.querySelector('.e-
active').appendChild(tag);
},
removeTag: function() {

this.$refs.listviewInstance.$el.ej2_instances[0].selectedLI.children[1].remo
ve();
}
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/data-binding/remote-data-cs2" %}

Show items count in group header in Vue Listview component

The ListView component supports wrapping list items into a group based on the category. The category of each list item can be mapped with groupBy field of the data source. You can display grouped list items count in the list-header using the group header template. Refer to the following code sample to display grouped list item count.

APP.VUE

```
<template>
  <div class="control-section">
    <div id = 'flat-list'>
    <!-- ListView element -->
    <ejs-listview id='List' ref='list' :dataSource='data' :fields='fields'
:template="demotemplate" :groupTemplate="grouptemplate" >
    </ejs-listview>
    </div>
  </div>
</template>
<style>
  .count{
    float:right;
  }
  #List {
      display: block;
      margin: auto;
      border: 1px solid;
      border-color: #ccc;
      border-color: rgba(0, 0, 0, 0.12);
      width: 60%;
  }
  #List .settings {
      height: auto;
  }
  #List .e-list-item {
      height: auto;
      padding: 0;
      cursor: pointer;
      box-sizing: border-box;
      border: 0.1px solid #ccc;
  }
  #List .e-list-header .e-text {
      font-family: sans-serif;
      font-size: 18px;
      line-height: 26px;
  }
  #List #content {
    margin: 3px;
  }
  #List .name {
      font-size: 12px;
      line-height: 25px;
  }
  #info {
    line-height: 20px;
    font-size: 12px;
  }
  #postImg {
     margin: 9px;
 }
  #postContainer {
  width: inherit;
  margin: auto;
  display: inline-flex;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
var demoVue = Vue.component("demo", {
  template: `<div class="settings">
              <div id="postContainer">
                  <div id="postImg">
                      <img :src='data.image'
style="height:35px;width:35px;border-radius:50%; border: 1px solid #ccc;"
/></div>
                  <div id="content">
                      <div class="name">{{data.Name}}</div>
                      <div id="info">{{data.contact}}</div>
                  </div>
              </div>
          </div>`,
  data() {
    return {
      data: {}
    };
  }
});
var tempVue = Vue.component("demo", {
  template: `<div><span class="category">{{data.items[0].category}}</span>
<span class="count"> {{data.items.length}} Item(s)</span></div>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data: function() {
    return {
      data: [
        { Name: 'Nancy', contact:'(206) 555-985774', id: '1', image:
'https://ej2.syncfusion.com/demos/src/grid/images/1.png',  category:
'Experience'},
        { Name: 'Janet', contact: '(206) 555-3412', id: '2', image:
'https://ej2.syncfusion.com/demos/src/grid/images/3.png', category:
'Fresher' },
        { Name: 'Margaret', contact:'(206) 555-8122', id:'4', image:
'https://ej2.syncfusion.com/demos/src/grid/images/4.png', category:
'Experience' },
        { Name: 'Andrew ', contact:'(206) 555-9482', id: '5', image:
'https://ej2.syncfusion.com/demos/src/grid/images/2.png', category:
'Experience'},
        { Name: 'Steven', contact:'(71) 555-4848', id: '6', image:
'https://ej2.syncfusion.com/demos/src/grid/images/5.png', category:
'Fresher' },
        { Name: 'Michael', contact:'(71) 555-7773', id: '7', image:
'https://ej2.syncfusion.com/demos/src/grid/images/6.png', category:
'Experience' },
        { Name: 'Robert', contact:'(71) 555-5598', id: '8', image:
'https://ej2.syncfusion.com/demos/src/grid/images/7.png', category:
'Fresher' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        { Name: 'Laura', contact:'(206) 555-1189', id: '9', image:
'https://ej2.syncfusion.com/demos/src/grid/images/8.png', category:
'Experience' },
    ],
         fields: {text: 'Name', groupBy: 'category'},
       demotemplate: function () {
                return { template : demoVue}
            },
             grouptemplate: function () {
                return { template : tempVue}
            },
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/checklist-cs9" %}

Filter list items in the listview in Vue Listview component

The filtered data can be displayed in the ListView component depending upon on user inputs using the DataManager. Refer to the following steps to render the ListView with filtered data.

- Render a textbox to get input for filtering data.
- Render ListView with dataSource, and set the sortOrder property.
- Bind the keyup event for textbox to perform filtering operation. To filter list data, pass the list data source to the DataManager, manipulate the data using the executeLocal method, and then update filtered data as ListView dataSource.

APP.VUE

```
<template>
  <div class="control-section">
    <div id = 'sample'>
      <input class="e-input" type="text" id='textbox' ref='textboxEle'
placeholder="Filter" title="Type in a name" @keyup='onkeyup' />
      <!-- ListView element -->
      <ejs-listview id='list' ref='listObj' :dataSource='data'
:fields='fields' sortOrder='Ascending'>
      </ejs-listview>
    </div>
  </div>
</template>
<style>
#list {
  box-shadow: 0 1px 4px #ddd;
  border-bottom: 1px solid #ddd;
}
#sample {
  height: 220px;
  margin: 0 auto;
  display: table;
}
</style>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import { DataManager, Query, ODataV4Adaptor } from "@syncfusion/ej2-data";
Vue.use(ListViewPlugin);
export default {
  data: function() {
    return {
      data: [
        { text: "Hennessey Venom", id: "list-01" },
        { text: "Bugatti Chiron", id: "list-02" },
        { text: "Bugatti Veyron Super Sport", id: "list-03" },
        { text: "SSC Ultimate Aero", id: "list-04" },
        { text: "Koenigsegg CCR", id: "list-05" },
        { text: "McLaren F1", id: "list-06" }],
      fields: { text: "text", id: "id" },
    };
  },
  methods:{
   onkeyup: function(event){
      let keyupvalue = this.$refs.textboxEle.value;
      let data = new DataManager(this.data).executeLocal(new
Query().where("text", "startswith", keyupvalue, true));
      if (!keyupvalue) {
        this.$refs.listObj.dataSource = this.data.slice();
      } else {
        this.$refs.listObj.dataSource = data;
      }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/checklist-cs4" %}

In this demo, data has been filtered with starting character of the list items. You can also filter list items with ending character by passing the endswith in where clause instead of startswith.

Add and remove list items from listview in Vue Listview component

You can add or remove list items from the ListView component using the addItem and removeItem methods.

Refer to the following steps to add or remove a list item.

- Render the ListView with data source, and use the template property to append the delete icon for each list item. Also, bind the click event for the delete icon using the actionComplete handler.
- Render the Add Item button, and bind the click event. On the click event handler, pass data with random id to the addItem method to add a new list item on clicking the Add Item button.
- Bind the click handler to the delete icon created in step 1. Within the click event, remove the list item by passing the delete icon list item to removeItem method.

APP.VUE

```
<template>
```

```
  <div class="control-section">
    <div id = 'flat-list'>
    <!-- ListView element -->
    <ejs-listview id='sample-list-flat' ref='list' :dataSource='data'
:fields='fields' :template='datatemplate' :actionComplete='onComplete' >
    </ejs-listview>
    </div>
   <ejs-button id="btn" v-on:click.native="onClick">Add Item</ejs-button>
  </div>
</template>
<style>
 #sample-list-flat {
  margin: 40px auto;
  max-width: 500px;
}
#btn {
  margin:40px auto;
  display:block;
}
/* csslint ignore:start */
@font-face {
  font-family: "e-icon";
  src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tSfIAAAEoAAAAVmNtYXDnEOdVAAABiAAAADZnbHlmXOn
iGAAAAcgAAAFAaGVhZBC1AhkAAADQAAAANmhoZWEIUQQDAAAArAAAACRobXR4CAAAAAAAAYAAAAA
IbG9jYQCgAAAAAAHAAAAABm1heHABDgCYAAABCAAAACBuYW1lv4Bt4QAAAwgAAAIZcG9zdJx8QW4
AAAUkAAAAOwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAgABAAAAAQAApWcDV18
PPPUACwQAAAAAANbRXpQAAAAA1tFelAAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAACAIwAAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAAQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAAEAAAA
AAAAAoAAAAAIAAAAAA/QD9AALAIsAAAEHFwcnByc3JzcXNwUfHz8fLx8PHgLuhIRrg4NrhIRrg4P
9iQECAwQGBwcJCwsMDQ4PDxEREhMUFBUWFhcXFxkYGRkaGhkZGBkXFxcWFhUUFBMSEREPDw4NDAs
LCQcHBgQDAgEBAgMEBgcHCQsLDA0ODw8RERITFBQVFhYXFxcZGBkZGhoZGRgZFxcXFhYVFBQTEhE
RDw8ODQwLCwkHBwYEAwICg4OGa4SEa4ODaoCE7hoZGRgZFxcXFhYVFBQTEhERDw8ODQwLCwkHBwY
EAwIBAQIDBAYHBwkLCwwNDg8PERESExQUFRYWFxcXGRgZGRoaGRkYGRcXFxYWFRQUExIREQ8PDg0
MCwsJBwcGBAMCAQECAwQGBwcJCwsMDQ4PDxEREhMUFBUWFhcXFxkYGRkAAAASAN4AAQAAAAAAAAA
BAAAAAQAAAAAAAQAGAAEAAQAAAAAAAgAHAAcAAQAAAAAAwAGAA4AAQAAAAAABAAGABQAAQAAAAA
ABQALABoAAQAAAAAABgAGACUAAQAAAAAACgAsACsAAQAAAAAACwASAFcAAwABBAkAAAACAGkAAwA
BBAkAAQAMAGsAAwABBAkAAgAOAHcAAwABBAkAAwAMAIUAAwABBAkABAAMAJEAAwABBAkABQAWAJ0
AAwABBAkABgAMALMAAwABBAkACgBYAL8AAwABBAkACwAkARcgZGVsZXRlUmVndWxhcmRlbGV0ZWR
lbGV0ZVZlcnNpb24gMS4wZGVsZXRlRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXR
ybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABkAGUAbABlAHQAZQBSAGUAZwB1AGwAYQByAGQ
AZQBsAGUAdABlAGQAZQBsAGUAdABlAFYAZQByAHMAaQBvAG4AIAAxAC4AMABkAGUAbABlAHQAZQB
GAG8AbgB0ACAAZwBlAG4AZQByAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGk
AbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwB
uAC4AYwBvAG0AAAAAgAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAQIBAwARY2lyY2x
lLWNsb3NlLS0tMDIAAAA=)
    format("truetype");
  font-weight: normal;
  font-style: normal;
}
/* csslint ignore:end */
#sample-list-flat .delete-icon::after {
  font-family: "e-icon";
  content: "\e700";
  float: right;
```

```
  cursor: pointer;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ListViewPlugin);
Vue.use(ButtonPlugin);
var demoVue = Vue.component("demo", {
  template: `<div class='text-content'> {{data.text}} <span class = 'delete-
icon'></span> </div>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data: function() {
    return {
        data: [
          { text: "Hennessey Venom", id: "1", icon: "delete-icon" },
          { text: "Bugatti Chiron", id: "2", icon: "delete-icon" },
          { text: "Bugatti Veyron Super Sport", id: "3", icon: "delete-icon"
},
          { text: "Aston Martin One- 77", id: "4", icon: "delete-icon" },
          { text: "Jaguar XJ220", id: "list-5", icon: "delete-icon" },
          { text: "McLaren P1", id: "6", icon: "delete-icon" }
        ],
        fields: { text: 'text', iconCSS: 'icon' },
        datatemplate: function () {
                return { template : demoVue}
            }
    };
  },
   methods: {
        deleteItem: function(args){
            args.stopPropagation();
            let liItem = args.target.parentElement.parentElement;
            this.$refs.list.removeItem(liItem);
            this.onComplete();
        },
        onComplete: function(args) {
            let iconEle = document.getElementsByClassName("delete-icon");
            var _this =this;
            //Event handler to bind the click event for delete icon
            Array.from(iconEle).forEach(function(element) {
                element.addEventListener("click",
_this.deleteItem.bind(_this));
            });
        },
        onClick: function(){
            let data = {
                    text: "Koenigsegg - " + (Math.random() *
1000).toFixed(0),
                    id: (Math.random() * 1000).toFixed(0).toString(),
```

```
                    icon: "delete-icon"
                };
            this.$refs.list.addItem([data]);
        }
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/add-item-cs1" %}

Trace all events in listview in Vue Listview component

The ListView component triggers events based on its actions. The events can be used as extension points to perform custom operations. Refer to the following steps to trace the ListView events:

1. Render the ListView with dataSource, and bind the actionBegin, actionComplete, and select events.
2. Perform custom operations in actionBegin, actionComplete, and select events.
3. Provide event log details for actionBegin and actionComplete events, and they will be displayed in the event trace panel when the ListView action starts and the dataSource bound successfully.
4. Get the selected item details from the SelectEventArgs in the select event, and display the selected list item text in the event trace panel while selecting list items.
5. Use clear button to remove event trace information.

APP.VUE

```
<template>
  <div id="sample">
    <div class="content-wrapper">
    <!-- ListView element -->
      <ejs-listview id='listview-def' ref='listviewInstance'
:dataSource='data' width=250 :actionBegin='onBegin'
:actionComplete='onComplete' :select='onSelect'>
      </ejs-listview>
    </div>
    <div id="list_event">
      <h4><b>Event Trace</b></h4>
      <div id="evt">
        <div class="eventarea" style="height:273px;overflow: auto">
          <span ref='EventLogEle' class="EventLog" id="EventLog"
style="word-break: normal;"></span>
        </div>
        <div class="evtbtn">
          <ejs-button id="clear" @click.native='onClick'>Clear</ejs-button>
        </div>
      </div>
    </div>
  </div>
</template>
<style>
#EventLog b {
  color: #388e3c;
}
```

```
#listview-def {
  border: 1px solid #dcdcdc;
}
.content-wrapper {
  padding-left: 40px;
  padding-top: 36px;
}
.evtbtn {
  margin-top: 40px;
  margin-left: 70px;
}
/* csslint ignore:start */
hr {
  margin-top: 6px !important;
  margin-bottom: 6px !important;
}
/* csslint ignore:end */
#evt {
  border: 1px solid #dcdcdc;
  padding: 10px;
  min-width: 10px;
}
#sample {
  display: inline-flex;
}
.eventarea {
  min-width: 250px;
}
#list_event {
  margin-top: -25px;
  padding-left:40px;
  min-width: 200px;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import {ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ListViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data: function() {
    return {
      data: [
  { text: "Hennessey Venom", id: "list-01" },
  { text: "Bugatti Chiron", id: "list-02" },
  { text: "Bugatti Veyron Super Sport", id: "list-03" },
  { text: "SSC Ultimate Aero", id: "list-04" },
  { text: "Koenigsegg CCR", id: "list-05" },
  { text: "McLaren F1", id: "list-06" },
  { text: "Aston Martin One- 77", id: "list-07" },
  { text: "Jaguar XJ220", id: "list-08" },
  { text: "McLaren P1", id: "list-09" },
  { text: "Ferrari LaFerrari", id: "list-10" }
]
    };
  },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  methods:{
    onBegin: function(){
     this.appendElement("<b>actionBegin </b> event is triggered<hr>");
    },
    onComplete: function(){
      this.appendElement("<b>actionComplete</b> is triggered <hr>");
    },
    onSelect: function(args){
    this.appendElement(args.text + "<b>  is selected</b><hr>");
    },
    appendElement: function(html) {
     let span = document.createElement("span");
     span.innerHTML = html;
     let log = this.$refs.EventLogEle;
     log.insertBefore(span, log.firstChild);
    },
    onClick: function(event){
       this.$refs.EventLogEle.innerHTML = "";
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/checklist-cs10" %}

Create mobile contact layout from listview in Vue Listview component

You can customize the ListView using the template property. Refer to the following steps to customize ListView as mobile contact view with our ej2-avatar.

- Render the ListView with dataSource that has avatar data. You can set avatar data as either text or class names. Refer to the following codes.

`ts

dataSource: [

{

text: "Jenifer", contact: "(206) 555-985774", id: "1", avatar: "", pic: "pic01"

},

{

text: "Amenda", contact: "(206) 555-3412", id: "2", avatar: "A", pic: ""

}

];

`

- Set avatar classes in ListView template to customize contact icon. In the following codes, medium size avatar has been set using the class name e-avatar e-avatar-circle from data source.

Copyright © 2001 -2024 Syncfusion Inc.

`ts

<span :class="['e-avatar e-avatar-small e-avatar-circle']" v-if="data.avatar !== ''">{{data.avatar}}

<span :class="[data.pic + ' e-avatar e-avatar-small e-avatar-circle']" v-if="data.pic !== '' ">

{{data.text}}

`

Avatars can be set in different sizes in avatar classes. To know more about avatar classes, refer to Avatar.

- Sort the contact names using the sortOder property of ListView.
- Enable the showHeader property, and set the headerTitle as Contacts.

APP.VUE

```
<template>
    <div id="sample">
       <ejs-listview id='List' :dataSource='data' :fields='fields'
:template='template' width="350px" showHeader='true' headerTitle='Contacts'
sortOrder='Ascending'></ejs-listview>
    </div>
</template>
<style>
#List {
  margin: 0 auto;
  border: 1px solid #ccc;
}
#List .e-list-item {
  height: 60px;
  cursor: pointer;
}
#List .e-list-header .e-text {
  font-family: sans-serif;
  font-size: 18px;
  line-height: 16px;
}
#List #content {
  margin: 0;
}
#List .e-list-header{
  background: rgb(2, 120, 215);
  color: white;
}
#List #info,
#List .name {
  font-size: 14px;
  margin: 0 60px;
  line-height: 20px;
}
#List .name {
```

```
  padding-top: 8px;
  font-weight: 500;
}
.pic01 {
  background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/1.png");
}
.pic02 {
  background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/3.png");
}
.pic03 {
  background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/5.png");
}
.pic04 {
  background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/2.png");
}
#List .e-avatar {
  position: absolute;
  margin-top: 8px;
  font-size: 14px;
}
#List .e-list-item:nth-child(1) .e-avatar {
  background-color: #039be5;
}
#List .e-list-item:nth-child(2) .e-avatar {
  background-color: #e91e63;
}
#List .e-list-item:nth-child(6) .e-avatar {
  background-color: #009688;
}
#List .e-list-item:nth-child(8) .e-avatar {
  background-color: #0088;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
var demoVue = Vue.component("demo", {
  template: `
    <div class="settings">
        <span :class="['e-avatar e-avatar-circle']" v-if="data.avatar !==
''">{{data.avatar}}</span>
        <span :class="[data.pic + ' e-avatar e-avatar-circle']" v-
if="data.pic !== '' "> </span>
        <div id="content">
        <div class="name">{{data.text}}</div>
        <div id="info">{{data.contact}}</div>
        </div>
    </div>`,
  data() {
    return {
      data: {}
    };
```

```
  }
});
export default {
  data: function() {
    return {
      data: [
  {
    text: "Jenifer",
    contact: "(206) 555-985774",
    id: "1",
    avatar: "",
    pic: "pic01"
  },
  { text: "Amenda", contact: "(206) 555-3412", id: "2", avatar: "A", pic: ""
},
  {
    text: "Isabella",
    contact: "(206) 555-8122",
    id: "4",
    avatar: "",
    pic: "pic02"
  },
  {
    text: "William ",
    contact: "(206) 555-9482",
    id: "5",
    avatar: "W",
    pic: ""
  },
  {
    text: "Jacob",
    contact: "(71) 555-4848",
    id: "6",
    avatar: "",
    pic: "pic04"
  },
  { text: "Matthew", contact: "(71) 555-7773", id: "7", avatar: "M", pic: ""
},
  {
    text: "Oliver",
    contact: "(71) 555-5598",
    id: "8",
    avatar: "",
    pic: "pic03"
  },
  {
    text: "Charlotte",
    contact: "(206) 555-1189",
    id: "9",
    avatar: "C",
    pic: ""
  }
],
    fields: {text: 'text'},
    template: function () {
                return { template : demoVue};
            }
```

```
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/avatar-cs1" %}

Display spinner until list items are loaded in Vue Listview component

The features of the ListView component such as remote data-binding take more time to fetch data from corresponding dataSource/remote URL. In this case, you can use EJ2 Spinner to enhance the appearance of the UI. This section explains how to load a spinner component to groom the appearance.

Refer to the following code sample to render the spinner component.

`ts

createSpinner({

target: this.spinnerEle.nativeElement

});

showSpinner(this.spinnerEle.nativeElement);

`

Here, the data is fetched from Northwind Service URL; it takes a few seconds to load the data. To enhance the UI, the spinner component has been rendered initially. After the data is loaded from remote URL, the spinner component will be hidden in ListView actionComplete event.

APP.VUE

```
<template>
 <div class="control-section">
    <div id="flat-list">
    <!-- ListView element -->
    <ejs-listview id='sample-list' :dataSource='data' :query='query'
:fields='fields' :headerTitle='headerTitle' showHeader='true'
:actionComplete='onComplete'></ejs-listview>
    </div>
    <div ref='spinnerEle' id="spinner" ></div>
  </div>
</template>
<style>
#sample-list {
    border: 1px solid #dddddd;
    border-radius: 3px;
    margin: auto;
}
#flat-list {
    width: 50%;
    padding: 10px;
    margin: auto;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
```

```
import { DataManager, Query } from '@syncfusion/ej2-data';
import { createSpinner, showSpinner, setSpinner } from '@syncfusion/ej2-vue-
popups';
Vue.use(ListViewPlugin);
export default {
  data: function() {
    return {
      data: new DataManager({
        url: 'https://services.syncfusion.com/js/production/api/',
        crossDomain: true
      }),
      query: new
Query().from("ListView").select("EmployeeID,FirstName").take(10),
      fields:  { id: "EmployeeID", text: "FirstName" },
      headerTitle: 'Employees',
    };
  },
  mounted: function(){
     createSpinner({ target: this.$refs.spinnerEle });
    showSpinner(this.$refs.spinnerEle);
  },
  methods: {
    onComplete: function(){
      this.$refs.spinnerEle.style.display = "none";
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/data-binding/remote-data-cs3" %}

Hide checkbox in listview in Vue Listview component

The checkbox of the any list item can be hidden by using[htmlAttributes](https://ej2.syncfusion.com/vue/documentation/api/list-view/#h tmlattributes) of fields object. With the help of htmlAttributes we can add unique class to each list item that will be rendered from the data source, from the CSS class we can hide the checkbox of the list item.

In this sample, we had hidden the multiple leaf node of nested list. The e-checkbox-hidden class has been added in the data source where the checkbox needs to be hidden. Refer the below snippet for simple data source.

`ts

{

'text': 'New York',

'id': '3002',

'category': 'USA',

'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' }

}

`

Copyright © 2001 -2024 Syncfusion Inc.

Even though we have hidden the checkbox the functionality will be same for the list item which might affect the getSelectedItems method. So, to counteract that we will follow certain logic in the select event. The Logic here is to remove the e-active class from the other checkbox hidden list item which will be added when we select on that item and retain e-active on currently selected item.

In this process we will exclude the visible checkbox list items and only consider the hidden checkbox items.

APP.VUE

```
<template>
  <div class="control-section">
    <div id = 'flat-list'>
      <!-- ListView element -->
       <ejs-listview id='folderCheckbox' ref='listviewInstance'
:dataSource='data' :fields='fields' sortOrder='Ascending' showHeader=true
headerTitle='Mixed Leaf Checkbox Hidden List' showCheckBox=true
:select='onSelect'>
       </ejs-listview>
    </div>
  </div>
</template>
<style>
  #folderCheckbox {
    border: 1px solid #dddddd;
    border-radius: 3px;
    margin: 5px;
    width: 400px;
  }
  #folderCheckbox .e-checkbox-hidden .e-checkbox-wrapper {
    visibility: hidden;
  }
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
export default {
  data: function() {
    return {
      data: [
    {
        'text': 'Asia',
        'id': '01',
        'category': 'Continent',
        'child': [{
            'text': 'India',
            'id': '1',
            'category': 'Asia',
            'child': [{
                'text': 'Delhi',
                'id': '1001',
                'category': 'India',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
            {
```

```
                'text': 'Kashmir',
                'id': '1002',
                'category': 'India',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
            {
                'text': 'Goa',
                'id': '1003',
                'category': 'India',
                'htmlAttributes': { 'class': 'e-file' },
            },
            ]
        },
        {
            'text': 'China',
            'id': '2',
            'category': 'Asia',
            'child': [{
                'text': 'Zhejiang',
                'id': '2001',
                'category': 'China',
                'htmlAttributes': { 'class': 'e-file' },
            },
            {
                'text': 'Hunan',
                'id': '2002',
                'category': 'China',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
            {
                'text': 'Shandong',
                'id': '2003',
                'category': 'China',
                'htmlAttributes': { 'class': 'e-file' },
            }]
        }]
    },
    {
        'text': 'North America',
        'id': '02',
        'category': 'Continent',
        'child': [{
            'text': 'USA',
            'id': '3',
            'category': 'North America',
            'child': [{
                'text': 'California',
                'id': '3001',
                'category': 'USA',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
            {
                'text': 'New York',
                'id': '3002',
                'category': 'USA',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
```

```
            {
                'text': 'Florida',
                'id': '3003',
                'category': 'USA',
                'htmlAttributes': { 'class': 'e-file' },
            }]
        },
        {
            'text': 'Canada',
            'id': '4',
            'category': 'North America',
            'child': [{
                'text': 'Ontario',
                'id': '4001',
                'category': 'Canada',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
            {
                'text': 'Alberta',
                'id': '4002',
                'category': 'Canada',
                'htmlAttributes': { 'class': 'e-file' },
            },
            {
                'text': 'Manitoba',
                'id': '4003',
                'category': 'Canada',
                'htmlAttributes': { 'class': 'e-file' },
            }]
        }]
    },
    {
        'text': 'Europe',
        'id': '03',
        'category': 'Continent',
        'child': [{
            'text': 'Germany',
            'id': '5',
            'category': 'Europe',
            'child': [{
                'text': 'Berlin',
                'id': '5001',
                'category': 'Germany',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
            {
                'text': 'Bavaria',
                'id': '5002',
                'category': 'Germany',
                'htmlAttributes': { 'class': 'e-file' },
            },
            {
                'text': 'Hesse',
                'id': '5003',
                'category': 'Germany',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            }]
```

```
        }, {
            'text': 'France',
            'id': '6',
            'category': 'Europe',
            'child': [{
                'text': 'Paris',
                'id': '6001',
                'category': 'France',
                'htmlAttributes': { 'class': 'e-file' },
            },
            {
                'text': 'Lyon',
                'id': '6002',
                'category': 'France',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
            {
                'text': 'Marseille',
                'id': '6003',
                'category': 'France',
                'htmlAttributes': { 'class': 'e-file' },
            }]
        }]
    },
    {
        'text': 'Australia',
        'id': '04',
        'category': 'Continent',
        'child': [{
            'text': 'Australia',
            'id': '7',
            'category': 'Australia',
            'child': [{
                'text': 'Sydney',
                'id': '7001',
                'category': 'Australia',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
            {
                'text': 'Melbourne',
                'id': '7002',
                'category': 'Australia',
                'htmlAttributes': { 'class': 'e-file' },
            },
            {
                'text': 'Brisbane',
                'id': '7003',
                'category': 'Australia',
                'htmlAttributes': { 'class': 'e-file' },
            }]
        }, {
            'text': 'New Zealand',
            'id': '8',
            'category': 'Australia',
            'child': [{
                'text': 'Milford Sound',
                'id': '8001',
```

```
                'category': 'New Zealand',
                'htmlAttributes': { 'class': 'e-file' },
            },
            {
                'text': 'Tongariro National Park',
                'id': '8002',
                'category': 'New Zealand',
                'htmlAttributes': { 'class': 'e-file' },
            },
            {
                'text': 'Fiordland National Park',
                'id': '8003',
                'category': 'New Zealand',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            }]
        }]
    },
    {
        'text': 'Africa',
        'id': '05',
        'category': 'Continent',
        'child': [{
            'text': 'Morocco',
            'id': '9',
            'category': 'Africa',
            'child': [{
                'text': 'Rabat',
                'id': '9001',
                'category': 'Morocco',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
            {
                'text': 'Toubkal',
                'id': '9002',
                'category': 'Morocco',
                'htmlAttributes': { 'class': 'e-file' },
            },
            {
                'text': 'Todgha Gorge',
                'id': '9003',
                'category': 'Morocco',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            }]
        }, {
            'text': 'South Africa',
            'id': '10',
            'category': 'Africa',
            'child': [{
                'text': 'Cape Town',
                'id': '10001',
                'category': 'South Africa',
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
            {
                'text': 'Pretoria',
                'id': '10002',
                'category': 'South Africa',
```

```
                'htmlAttributes': { 'class': 'e-file e-checkbox-hidden' },
            },
            {
                'text': 'Bloemfontein',
                'id': '10003',
                'category': 'South Africa',
                'htmlAttributes': { 'class': 'e-file' },
            }]
        }]
    }
],
    fields: { tooltip: "text" },
    };
  },
  methods:{
   onSelect: function(args){
      let normalElements =
Array.prototype.slice.call(this.$refs.listviewInstance.$el.ej2_instances[0].
curUL.getElementsByClassName('e-checkbox-hidden'));
        // Looping through all the selected element and removing e-active
class
        // to avoid behaviour interference  with getSelectedItems method
        normalElements.forEach((element) => {
            element.classList.remove('e-active');
        });
        // Finally adding e-active class to currently selected item except
checkbox item.
        // because if it is checkbox item their actions will taken care from
the source side itself.
        if (args.item.classList.contains('e-checkbox-hidden')) {
            args.item.classList.add('e-active');
        }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/checklist-cs6" %}

Manipulate listview as grid layout in Vue Listview component

In Listview, list items can be rendered in grid layout with following data manipulations.

- Add Item
- Remove Item
- Sort Items
- Filter Items

Grid Layout

In this section, we will discuss about rendering of list items in grid layout.

- Initialize and render ListView with dataSource which will render list items in list layout.
- Now, add the below CSS to list item. This will make list items to render in grid layout

`

element .e-list-item {
height: 100px;

width: 100px;

float: left;

}

`

In the below sample, we have rendered List items in grid layout.

APP.VUE

```
<template>
  <div id="sample">
    <ejs-listview id='element' :dataSource='data' :template='listTemplate'>
    </ejs-listview>
  </div>
</template>
<style>
#element {
    display: block;
    max-width: 303px;
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
}
#element .e-list-item {
    height: 100px;
    width: 100px;
    float: left;
    padding: 0;
}
#listImage {
    width: 55px;
    height: 55px;
    margin-left: 20px;
    margin-top: 20px;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
var listVue = Vue.component("demo", {
  template: `<img id="listImage"
src="https://ej2.syncfusion.com/documentation/code-snippet/listview/grid-
layout-cs1/apple.png" alt="apple" />`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
```

```
  data: function() {
          return {
    data: [1, 2, 3, 4, 5, 6, 7, 8, 9],
    listTemplate: function () {
      return { template : listVue}
    },
  };
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/grid-layout-cs1" %}

Data manipulation

In this section, we will discuss about ListView data manipulations.

Add Item

We can add list item using addItem API. This will accept array of data as argument.

`ts

this.$refs.listViewInstance.addItem([{text: 'Apricot', id: '32'}]);

`

In the below sample, you can add new fruit item by clicking add button which will open dialog box with fruit name and image URL text box. After entering the item details, click the add button. This will add your new fruit item.

Remove item

We can remove list item using removeItem API. This will accept fields with id or list item element as argument.

`ts

this.$refs.listViewInstance.removeItem({id: '32'});

`

In the below sample, you can remove fruit by hovering the fruit item which will show delete button and click that delete button to delete that fruit from your list.

Sort Items

Listview can be sorted either in Ascending or Descending order. To enable sorting in your ListView, set sortOrder as Ascending or Descending.

`ts

<ejs-listview sortOrder='Ascending'></ejs-listview>

`

We can also set sorting after component initialization.

`ts

this.$refs.listViewInstance.sortOrder = 'Ascending'

`

Copyright © 2001 -2024 Syncfusion Inc.

In the below sample, we have sorted fruits in Ascending order. To sort it in descending, click on sort order icon and vice versa.

Filter Items

Listview data can be filtered with the help of dataManager. After filtering the data, update ListView dataSource with filtered data.

`ts

let value = this.$refs.textboxEle.value; //input text box value

let filteredData = new DataManager(this.listdata).executeLocal(

new Query().where("text", "startswith", value, true)

);

listViewInstance.dataSource = filteredData;

`

In the below sample, we can filter fruit items with the help of search text box. This will filter fruit items based on your input. Here we used startswith of input text to filter data in DataManager.

APP.VUE

```
<template>
    <div id="sample">
        <div class="headerContainer">
            <div class="e-input-group">
                <input id="search" ref='searchEle' class="e-input"
type="text" placeholder="Search fruits" @keyup='onKeyUp'/>
                <span class="e-input-group-icon e-input-search"></span>
            </div>
            <ejs-button id="sort" class="e-control e-btn e-small e-round e-
primary e-icon-btn" v-on:click='sortItems' title="Sort fruits" data-
ripple="true">
                <span class="e-btn-icon e-icons e-sort-icon-
ascending"></span>
            </ejs-button>
            <ejs-button id="add" class="e-control e-btn e-small e-round e-
primary e-icon-btn" v-on:click='addItem' title="Add fruit" data-
ripple="true">
                <span class="e-btn-icon e-icons e-add-icon"></span>
            </ejs-button>
            <ejs-dialog id="dialog" ref='dialogObj' width='300px'
:content='content' :visible='visible' header='Add Fruit'
showCloseIcon='true' :buttons='addButtons' ></ejs-dialog>
        </div>
            <ejs-listview id='element' ref='listViewInstance'
:dataSource='fruitsdata' :template='listTemplate' sortOrder='Ascending'
:actionComplete='wireEvents'>
            </ejs-listview>
    </div>
</template>
<style>
#listImage {
    width: 55px;
    height: 55px;
```

```
    margin-left: 25px;
}
#sample {
    max-width: 440px;
    margin: auto;
    box-shadow: 0 3px 6px lightgray;
}
.headerContainer {
    height: 48px;
    line-height: 48px;
    background: rgb(2, 120, 215);
    color: white;
    margin-bottom: 3px;
}
.headerContainer .e-input-group {
    margin-left: 20px;
    width: 200px;
    background: white;
    height: 31px;
}
.headerContainer #search {
    height: 21px;
    margin-left: 10px;
}
#listDialog .input_name {
    margin-bottom: 20px;
}
.headerContainer #add,
.headerContainer #sort {
    float: right;
    margin-right: 15px;
    margin-top: 7px;
    background: white;
    color: black
}
.headerContainer .e-input-search::before {
    font-family: 'e-icons';
    content: '\e961';
    margin-top: 3px;
}
.headerContainer .e-input-group .e-input-group-icon.e-input-search {
    padding: 0 10px 0 10px;
}
#element .e-list-item {
    height: 110px;
    width: 110px;
    float: left;
    padding: 0;
    position: relative;
    user-select: none;
}
#element .e-delete-btn {
    float: right;
    visibility: hidden;
    margin-top: -10px;
}
#element .e-delete-btn.e-btn.e-small.e-round {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    width: 2em;
    height: 2em;
}
#element .e-btn.e-small.e-round .e-btn-icon.delete-icon {
    font-size: 9px;
}
#element .e-list-item:hover .e-delete-btn {
    visibility: visible;
    background: red;
    border-radius: 50%;
}
#element .fruits {
    height: inherit;
    width: inherit;
    padding: 10px 0 10px 0;
}
#element .fruitName {
    text-align: center;
}
.headerContainer .e-add-icon::before {
    content: '\e823';
}
#element .delete-icon::before {
    content: '\e7fc';
    color: white;
}
.headerContainer .e-sort-icon-ascending::before {
    content: '\e840';
}
.headerContainer .e-sort-icon-descending::before {
    content: '\e83f';
}
.e-dialog.e-control.e-popup.e-popup-open {
  max-height:361px !important;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import { DialogPlugin } from '@syncfusion/ej2-vue-popups';
import { closest, enableRipple } from '@syncfusion/ej2-base';
import { DataManager, Query } from "@syncfusion/ej2-data";
enableRipple(true);
Vue.use(ListViewPlugin);
Vue.use(DialogPlugin);
var listVue = Vue.component("demo", {
  template: `<div class="fruits"><div class="first"><img id="listImage"
:src="data.imgUrl" alt="fruit" /><ejs-button class="delete e-control e-btn
e-small e-round e-delete-btn e-primary e-icon-btn" data-ripple="true"><span
class="e-btn-icon e-icons delete-icon"></span></ejs-button></div><div
class="fruitName">{{data.text}}</div></div>`,
  data() {
    return {
      data: {}
    };
  }
});
```

Copyright © 2001 -2024 Syncfusion Inc.

```
var dialogVue = Vue.component("demo", {
  template: `<div id="listDialog"><div class="input_name"><label
for="name">Fruit Name: </label><input id="name" class="e-input" type="text"
placeholder="Enter fruit name"/></div><div><label for="imgurl">Fruit Image:
</label><input id="imgurl" class="e-input" type="text" placeholder="Enter
image url"/></div></div>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data: function() {
        return {
    fruitsdata: [
    { text: "Date", id: "1", imgUrl:
"https://ej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/dates.jpg" },
    { text: "Fig", id: "2", imgUrl:
"https://ej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/fig.jpg" },
    { text: "Apple", id: "3", imgUrl:
"https://ej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/apple.png" },
    { text: "Apricot", id: "4", imgUrl:
"https://ej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/apricot.jpg" },
    { text: "Grape", id: "5", imgUrl:
"https://ej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/grape.jpg" },
    { text: "Strawberry", id: "6", imgUrl:
"https://ej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/strawberry.jpg" },
    { text: "Pineapple", id: "7", imgUrl:
"https://ej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/pineapple.jpg" },
    { text: "Melon", id: "8", imgUrl:
"https://ej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/melon.jpg" },
    { text: "Lemon", id: "9", imgUrl:
"https://ej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/lemon.jpg" },
    { text: "Cherry", id: "10", imgUrl:
"https://ej2.syncfusion.com/documentation/code-
snippet/listview/manipulation-cs1/cherry.jpg" }
],
    listTemplate: function () {
      return { template : listVue};
    },
    visible: false,
     content: function () {
      return { template : dialogVue};
    },
    addButtons: [{
        click: this.dlgButtonClick, buttonModel: { content: 'Add',
isPrimary: true }
```

```
    }]
  };
},
methods: {
  wireEvents: function() {
    Array.prototype.forEach.call(document.getElementsByClassName('e-delete-
btn'), (ele) => {
        ele.addEventListener('click', this.onDeleteBtnClick);
    });
    },
addItem: function() {
    (document.getElementById("name")).value = "";
    (document.getElementById("imgurl")).value = "";
    this.$refs.dialogObj.show();
},
sortItems: function() {
    let ele = document.getElementById("sort").firstElementChild;
    let des = ele.classList.contains('e-sort-icon-descending') ? true :
false;
    if (des) {
        ele.classList.remove('e-sort-icon-descending');
        ele.classList.add('e-sort-icon-ascending');
        this.$refs.listViewInstance.sortOrder = 'Ascending';
    } else {
        ele.classList.remove('e-sort-icon-ascending');
        ele.classList.add('e-sort-icon-descending');
        this.$refs.listViewInstance.sortOrder = 'Descending'
    }
    this.$refs.listViewInstance.dataBind();
    this.wireEvents();
},
onKeyUp: function(e) {
    let value = this.$refs.searchEle.value;
    let data = new DataManager(this.fruitsdata).executeLocal(
        new Query().where("text", "startswith", value, true)
    );
    if (!value) {
        this.$refs.listViewInstance.dataSource = this.fruitsdata.slice();
    } else {
        this.$refs.listViewInstance.dataSource = data;
        this.$refs.listViewInstance.dataBind();
    }
},
onDeleteBtnClick: function(e) {
    e.stopPropagation();
    let li = closest(e.currentTarget, '.e-list-item');
    let data = this.$refs.listViewInstance.findItem(li);
    this.$refs.listViewInstance.removeItem(data);
    new DataManager(this.fruitsdata).remove('id', { id: data.id });
},
 dlgButtonClick: function() {
    let name = (document.getElementById("name")).value;
    let url = (document.getElementById("imgurl")).value;
    let id = Math.random() * 10000;
    this.$refs.listViewInstance.addItem([{ text: name, id: id, imgUrl: url
}]);
    this.fruitsdata.push({ text: name, id: id, imgUrl: url });
```

Copyright © 2001 -2024 Syncfusion Inc.

```
this.$refs.listViewInstance.$el.ej2_instances[0].element.querySelector('[dat
a-uid="'+ id + '"]').getElementsByClassName('e-delete-
btn')[0].addEventListener('click', this.onDeleteBtnClick);
    this.$refs.dialogObj.hide();
}
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/grid-layout-cs2" %}

Drag and drop list items in Vue Listview component

In ListView component, we don't have drag and drop support. But we can achieve this requirement using TreeView component with ListView appearance.

Drag and Drop in TreeView component was enabled by setting allowDragAndDrop to true.

`ts

<ejs-treeview id='element' :fields='fields' allowDragAndDrop='true'></ejs-treeview>

`

The TreeView component is used to represent hierarchical data in a tree like structure. So, list items in TreeView can be dropped to child of target element. we can prevent this behaviour by cancelling the nodeDragStop and nodeDragging events.

`ts

<ejs-treeview id='element' :fields='fields' allowDragAndDrop='true' :nodeDragging='onDragStop' :nodeDragStop='onDragStop'></ejs-treeview>

fields: { dataSource: data, id: 'id', text: 'text' },

onDragStop: function(args) {

//Block the Child Drop operation in TreeView

let draggingItem = document.getElementsByClassName("e-drop-in");

if (draggingItem.length == 1) {

draggingItem[0].classList.add('e-no-drop');

args.cancel = true;

}

}

`

In the below sample, we have rendered draggable list items.

APP.VUE

```
<template>
   <div id="sample">
    <ejs-treeview id='element' :fields='fields' allowDragAndDrop='true'
:nodeDragging='onDragStop' :nodeDragStop='onDragStop'></ejs-treeview>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
</template>
<style>
#sample {
    display: block;
    max-width: 280px;
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
}
#element.e-treeview .e-ul {
    padding: 0;
}
#element.e-treeview .e-list-item {
    padding: 0 16px;
}
#element.e-treeview .e-text-content {
    padding: 0;
}
#element.e-treeview .e-fullrow {
    height: 36px;
}
#element.e-treeview .e-list-text {
    line-height: 34px;
}
#element.e-treeview .e-list-item:last-child {
    margin-bottom: 9px;
}
#element.e-treeview .e-list-item:first-child {
    margin-top: 9px;
}
</style>
<script>
import Vue from "vue";
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data: function() {
         var  data =[
    { text: 'Hennessey Venom', id: 'list-01' },
    { text: 'Bugatti Chiron', id: 'list-02' },
    { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
    { text: 'SSC Ultimate Aero', id: 'list-04' },
    { text: 'Koenigsegg CCR', id: 'list-05' },
    { text: 'McLaren F1', id: 'list-06' },
    { text: 'Aston Martin One- 77', id: 'list-07' },
    { text: 'Jaguar XJ220', id: 'list-08' },
    { text: 'McLaren P1', id: 'list-09' },
    { text: 'Ferrari LaFerrari', id: 'list-10' },
];
  return {
   fields: { dataSource: data, id: 'id', text: 'text' },
    };
  },
  methods: {
    onDragStop:function(args) {
    //Block the Child Drop operation in TreeView
```

```
    let draggingItem = document.getElementsByClassName("e-drop-in");
    if (draggingItem.length == 1) {
        draggingItem[0].classList.add('e-no-drop');
        args.cancel = true;
    }
}
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/drag-drop-cs1" %}

Load html content via ajax in Vue Listview component

We can set external HTML page content as template for our ListView component by making use of AJAX call.

`ts

let ajax = new Ajax('./template.html', 'GET', false);

ajax.onSuccess = (e)=>{

this.template = e;

}

ajax.send();

`

In the below sample, we have rendered smartphone settings template from external HTML file.

APP.VUE

```
<template>
   <div id="sample">
    <ejs-listview id='element' :dataSource='data' headerTitle='Settings'
showHeader='true' :template='template'>
    </ejs-listview>
  </div>
</template>
<style>
  #element {
    display: block;
    max-width: 300px;
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
}
#element .e-list-header {
    background: rgb(2, 120, 215);
    color: white;
    font-size: 19px;
    font-weight: 500;
}
#element.e-listview .e-list-item {
    padding: 10px 16px;
    height: 61px;
```

```
}
#element .e-list-item-header {
    font-size: 15px;
    font-weight: 500;
    line-height: 20px;
    height: 20px;
}
#element .e-list-content {
    line-height: 20px;
    height: 20px;
    font-size: 10px;
    font-style: italic;
    margin-top: -2px;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import { Ajax } from '@syncfusion/ej2-base';
Vue.use(ListViewPlugin);
export default {
  data: function() {
    return {
      data: [
    { name: 'Network & Internet', id: '0', description: 'Wi-Fi, mobile, data
usage, hotspot' },
    { name: 'Connected devices', id: '1', description: 'Bluetooth, cast,
NFC' },
    { name: 'Battery', id: '2', description: '18% -4h 12m left' },
    { name: 'Display', id: '3', description: 'Wallpaper, sleep, font size'
},
    { name: 'Sound', id: '4', description: 'Volume, vibration, Do Not
Disturb' },
    { name: 'Storage', id: '5', description: '52% used - 15.48 GB free' }
],
    template:{}
    };
  },
  created: function() {
    let ajax = new Ajax('./template.html', 'GET', false);
    ajax.onSuccess = (e)=>{
    this.template = e;
    };
    ajax.send();
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/ajax-cs1" %}

Render listview with hyper link navigation in Vue Listview component

We can use anchor tag along with href attribute in our ListView template property for navigation.

`ts

var listVue = Vue.component("demo", {

template: {{data.name}},

data() {

return {

data: {}

};

}

});

`

In the below sample, we have rendered ListView with search engines URL.

APP.VUE

```
<template>
   <div id="sample">
    <ejs-listview id='List' :dataSource='data' headerTitle='Search engines'
showHeader='true' :template='anchortemplate'>
    </ejs-listview>
  </div>
</template>
<style>
 #List {
        max-width: 300px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
    #List a {
      text-decoration: none;
    }
    #List .e-list-header {
        background: rgb(2, 120, 215);
        color: white;
        font-size: 19px;
        font-weight: 500;
    }
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
Vue.use(ListViewPlugin);
var listVue = Vue.component("demo", {
  template: `<a target='_blank' v-bind:href='data.url'>{{data.name}}</a>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data: function() {
       return {
```

```
      data: [
        {name: 'Google', url: 'https://www.google.com'},
        {name: 'Bing', url: 'https://www.bing.com' },
        {name: 'Yahoo', url: 'https://www.yahoo.com'},
        {name: 'Ask.com', url: 'https://www.ask.com'},
        {name: 'AOL.com', url: 'https://www.aol.com'},
    ],
    anchortemplate: function () {
        return { template : listVue};
    },
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/checklist-cs8" %}

Customize listview as chat window in Vue Listview component

ListView can be customized as chat window. To achieve that, use the ListView template property and Avatar component.

- The Listview template property is used to showcase the ListView as chat window.
- Avatar component is used to design the image of contact person.

Refer the below template code snippet for Template of chat window.

`ts

var demoVue = Vue.component("demo", {

template: `

{{data.text}}

{{data.contact}}

{{data.avatar}}

<span :class="[data.pic + ' img1 e-avatar e-avatar-circle']">

<div class='settings' v-else>

{{data.avatar}}

<span :class="[data.pic +' img2 e-avatar e-avatar-circle']">

{{data.text}}

{{data.contact}}

`,

data() {

return {

data: {}

};

}

});

`

Chat order in template

In ListView template, we have rendered the list items based on receiver and sender information from dataSource of listview.

Adding messages to chat window

- Use textbox to get message from user.
- Add the textbox message to ListView dataSource using addItem method.

`ts

btnClick: function() {

let value = this.$refs.textbox.value;

this.$refs.listObj.addItem([{ text: "Amenda", contact: value, id: "2", avatar: "A", pic: "", chat: "receiver" }]);

this.$refs.textbox.value = "";

}

`

APP.VUE

```
<template>
   <div class="control-section">
    <!-- ListView element -->
    <ejs-listview id='List' ref='listObj' :dataSource='data'
:headerTitle='headerTitle' :template='Template' :showHeader='true'
:fields='fields' :width='width'></ejs-listview>
  <div style="width: 350px;margin: 0 auto;"><input id="name" ref="textbox"
style="width: 275px" class="e-input" type="text" placeholder="Type your
message"/>
    <ejs-button id="btn" style="float:right" v-
on:click.native="btnClick">Send</ejs-button></div>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
#List {
  margin: 0 auto;
  border: 1px solid #ccc;
}
#List .e-list-item {
  height: auto;
  cursor: pointer;
}
#List .e-list-header .e-text {
  font-family: sans-serif;
  font-size: 18px;
  line-height: 26px;
}
#List #info,
#List .name {
  font-size: 11px;
  line-height: 20px;
}
#List .name {
  padding-top: 3px;
  font-weight: 500;
   padding-left:150px;
}
#List #info {
  float: right;
  margin-right:10px;
}
.pic01 {
  background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/1.png");
}
.pic02 {
  background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/3.png");
}
.pic03 {
  background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/5.png");
}
.pic04 {
  background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/2.png");
}
.img2.e-avatar {
  margin-left: 10px;
  margin-top: 7px !important;
  font-size: 13px;
}
#List #content1 {
  width: 200px;
  background-color: aliceblue;
  display: inline-block;
  margin: 5px;

}
#List #info1,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
#List .name1 {
  font-size: 11px;
  line-height: 20px;
  margin-left: 10px;
}
#List .name1 {
  padding-top: 3px;
  font-weight: 500;
}
#List #content {
    margin: 5px;
    width: 200px;
    margin-left: 90px;
    background-color: aliceblue;
    display:inline-block
}
#image {
    float: right;
    display: inline-block;
}
#image2 {
    float: left;
    display: inline-block;
}
.img1.e-avatar {
    margin-right: 10px;
    margin: 5px;
    font-size: 13px;
}
.e-listview .e-list-item {
  padding: 0px !important;
}
.e-listview .e-list-header{
  color: white !important;
}
.e-listview .e-list-header{
  background: rgb(2, 120, 215) !important;
}
#List.e-listview .e-list-item.e-hover {
  background-color: transparent;
}
</style>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ListViewPlugin);
Vue.use(ButtonPlugin);
var demoVue = Vue.component("demo", {
    template: `<div class='settings' v-if='data.chat!="receiver"'>
      <div id='content'>
      <div class='name'>{{data.text}}</div>
            <div id='info'>{{data.contact}}</div>
            </div>
            <div id='image' v-if='data.avatar!=""'><span class='e-avatar
img1 e-avatar-circle'>{{data.avatar}}</span></div>
```

```
            <div id='image'  v-if='data.avatar==""'><span :class="[data.pic
+ ' img1 e-avatar e-avatar-circle']"> </span></div>
            </div>
            <div class='settings' v-else>
            <div id='image2'  v-if='data.avatar!=""'><span class='e-avatar
img2 e-avatar-circle'>{{data.avatar}}</span></div>
            <div id='image2'  v-if='data.avatar==""'><span :class="[data.pic
+' img2 e-avatar e-avatar-circle']"> </span></div>
            <div id='content1'>
            <div class='name1'>{{data.text}}</div>
            <div id='info1'>{{data.contact}}</div>
            </div>
            </div>`,
    data() {
        return {
        data: {}
        };
    }
});
export default {
  data: function() {
    return {
     data:  [
        {
          text: "Jenifer",
          contact: "Hi",
          id: "1",
          avatar: "",
          pic: "pic01", chat: "sender"
        },
        { text: "Amenda", contact: "Hello", id: "2", avatar: "A", pic: "",
chat: "receiver" },
        {
          text: "Jenifer",
          contact: "What Knid of application going to launch",
          id: "4",
          avatar: "",
          pic: "pic02",chat: "sender"
        },
        {
          text: "Amenda ",
          contact: "A knid of Emergency broadcast App",
          id: "5",
          avatar: "A",
          pic: "", chat: "receiver"
        },
        {
          text: "Jacob",
          contact: "Can you please elaborate",
          id: "6",
          avatar: "",
          pic: "pic04",chat: "sender"
        },
      ],
      fields: { text: "Name" },
      width:'350px',
      headerTitle: 'Chat',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          Template: function(e) {
              return {
                  template: demoVue,
              };
          }
      };
    },
    methods: {
      btnClick: function() {
        let value = this.$refs.textbox.value;
        this.$refs.listObj.addItem([{ text: "Amenda", contact: value, id: "2",
avatar: "A", pic: "", chat: "receiver" }]);
        this.$refs.textbox.value = "";
      }
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/listview/chat-window-cs1" %}

Maps

Getting Started with the Vue Maps Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLILink to the Video and integrating the Syncfusion Vue Maps component

You can explore some useful features in the Maps component using the following video.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

Below is the list of minimum dependencies required to use the Maps.

`javascript

|-- @syncfusion/ej2-vue-maps

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-data

|-- @syncfusion/ej2-buttons

|-- @syncfusion/ej2-popups

|-- @syncfusion/ej2-splitbuttons

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-svg-base

|-- @syncfusion/ej2-maps

`

Copyright © 2001 -2024 Syncfusion Inc.

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

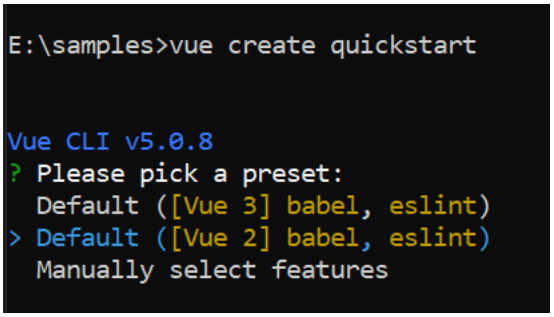
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Maps component as an example. Install the @syncfusion/ej2-vue-maps package by running the following command:

`bash

npm install @syncfusion/ej2-vue-maps --save

`

or

`bash

yarn add @syncfusion/ej2-vue-maps

`

Adding Syncfusion Vue Maps component

Follow the below steps to add the Vue Maps component:

1\. First, import and register the Maps component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { MapsComponent, LayerDirective, LayersDirective } from
'@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
export default {
components: {
'ejs-maps' : MapsComponent,
'e-layers' : LayersDirective,
'e-layer' : LayerDirective
},
data () {
return {
shapeData: world_map
}
}
}
</script>
```

2\. In the template section, define the Maps component.

~/SRC/APP.VUE

```
<template>
<div class="wrapper">
<ejs-maps id='maps'>
<e-layers>
<e-layer :shapeData='shapeData'></e-layer>
</e-layers>
</ejs-maps>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps id='maps'>
               <e-layers>
                    <e-layer :shapeData='shapeData'></e-layer>
                </e-layers>
            </ejs-maps>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </div>
    </div>
</template>
<script>
import { MapsComponent, LayersDirective, LayerDirective } from
'@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
export default {
  components: {
    'ejs-maps' : MapsComponent,
    'e-layers' : LayersDirective,
    'e-layer' : LayerDirective
  },
  data () {
    return {
        shapeData: world_map
    }
  }
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs105" %}

Module Injection

Maps component are segregated into individual feature-wise modules. In order to use a particular feature,

you need to inject its feature module using provide option. Find the modules available in Maps and its description as follows.

- Annotations - Inject this provider to use annotations feature.
- Bubble - Inject this provider to use bubble feature.
- DataLabel - Inject this provider to use data label feature.
- Highlight - Inject this provider to use highlight feature.

Copyright © 2001 -2024 Syncfusion Inc.

- Legend - Inject this provider to use legend feature.
- Marker - Inject this provider to use marker feature.
- MapsTooltip - Inject this provider to use tooltip series.
- NavigationLine - Inject this provider to use navigation lines feature.
- Selection - Inject this provider to use selection feature.
- Zoom - Inject this provider to use zooming and panning feature.

In the current application, we are going to modify the above basic Maps to visualize 2016 USA president election results.

For this application we are going to use tooltip, data label and legend features of the Maps. Now import the MapsTooltip, DataLabel and Legend modules from Maps package and inject it into the Maps component using provide option.

```
<template>
<div class="wrapper">
<ejs-maps id='maps'></ejs-maps>
</div>
</template>
<script>
import { MapsComponent, Legend, DataLabel, MapsTooltip } from '@syncfusion/ej2-vue-maps';
export default {
components: {
'ejs-maps': MapsComponent
}
data:function(){
return{ };
},
provide: {
maps: [Legend, DataLabel, MapsTooltip]
}
}
</script>
```

Render shapes from GeoJSON data

This section explains how to bind GeoJSON data to the map.

```javascript
```

Copyright © 2001 -2024 Syncfusion Inc.

let usMap: Object =

{

"type": "FeatureCollection",

"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },

"features": [

{ "type": "Feature", "properties": { "iso*31662*": "MA", "name": "Massachusetts", "admin": "United States of America" }, "geometry": { "type": "MultiPolygon", "coordinates": [[[[-70.801756294617277, 41.248076234530558]]]] }

}

]

//https://ej2.syncfusion.com/vue/documentation

};

`

Elements in the Maps will get rendered in the layers. So add a layer collection to the Maps by using layers property. Now bind the GeoJSON data to the shapeData property.

~/SRC/APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import { MapsComponent, LayerDirective, LayersDirective } from
'@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
export default {
components: {
    'ejs-maps': MapsComponent,
    'e-layer': LayerDirective,
    'e-layers': LayersDirective
},
data (){
    return{
        shapeData: world_map
    }
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs34" %}

Note: Refer the data values for world_map here.

Bind data source to map

The following properties in layers are used for binding data source to map.

- [dataSource]
- [shapeDataPath]
- [shapePropertyPath]

The dataSource property takes collection value as input. For example, the list of objects can be provided as input. This data is further used in tooltip, data label, bubble, legend and in color mapping.

The shapeDataPath property used to refer the data ID in dataSource. Where as, the shapePropertyPath property is used to refer the column name in shapeData to identify the shape. Both the properties are related to each other. When the values of the shapeDataPath property in the dataSource property and the value of shapePropertyPath in the shapeData property match, then the associated object from the dataSource is bound to the corresponding shape.

The JSON object "permanent and non-permanent countries in the UN security council" is used as data source below.

~/SRC/APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import { MapsComponent, LayerDirective, LayersDirective } from
'@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
export default {
components: {
    'ejs-maps': MapsComponent,
    'e-layer': LayerDirective,
    'e-layers': LayersDirective
},
data (){
    return{
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        shapeData: world_map,
        dataSource: [{  "Country": "China", "Membership": "Permanent"},
            {"Country": "France","Membership": "Permanent" },
            { "Country": "Russia","Membership": "Permanent"},
            {"Country": "Kazakhstan","Membership": "Non-Permanent"},
            { "Country": "Poland","Membership": "Non-Permanent"},
            {"Country": "Sweden","Membership": "Non-Permanent"}],
        shapePropertyPath: 'name',
        shapeDataPath: 'Country'
    }
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs35" %}

Note: Refer the data values for world_map here.

Apply Color Mapping

The Color Mapping feature supports customization of shape colors based on the underlying value of shape received from bounded data. Specify the field name from which the values have to be compared for the shapes in [colorValuePath] property in [shapeSettings].

Specify color and value in colorValuePath property. Here '#D84444' is specified for 'Permanent' and '#316DB5' is specified for 'Non-Permanent'.

~/SRC/APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import { MapsComponent, LayerDirective, LayersDirective } from
'@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
export default {
components: {
    'ejs-maps': MapsComponent,
    'e-layer': LayerDirective,
    'e-layers': LayersDirective
```

```
},
data (){
    return{
       shapeData: world_map,
        dataSource: [{  "Country": "China", "Membership": "Permanent"},
            {"Country": "France","Membership": "Permanent" },
            { "Country": "Russia","Membership": "Permanent"},
            {"Country": "Kazakhstan","Membership": "Non-Permanent"},
            { "Country": "Poland","Membership": "Non-Permanent"},
            {"Country": "Sweden","Membership": "Non-Permanent"}],
        shapePropertyPath: 'name',
        shapeDataPath: 'Country',
        shapeSettings: {
            colorValuePath: 'Membership',
                colorMapping: [
                {
                    value: 'Permanent', color: '#D84444'
                },
                {
                    value: 'Non-Permanent', color: '#316DB5'
                }]
        }
    }
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs36" %}

Add Title for Maps

You can add a title using titleSettings property to the map to provide quick

information to the user about the shapes rendered in the map.

~/SRC/APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps  :titleSettings='titleSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { MapsComponent, LayerDirective, LayersDirective } from
'@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
export default {
components: {
    'ejs-maps': MapsComponent,
    'e-layer': LayerDirective,
    'e-layers': LayersDirective
},
data (){
    return{
        titleSettings: {
            text: 'UN security council countries'
        },
        shapeData: world_map,
        dataSource: [{  "Country": "China", "Membership": "Permanent"},
            {"Country": "France","Membership": "Permanent" },
            { "Country": "Russia","Membership": "Permanent"},
            {"Country": "Kazakhstan","Membership": "Non-Permanent"},
            { "Country": "Poland","Membership": "Non-Permanent"},
            {"Country": "Sweden","Membership": "Non-Permanent"}],
        shapePropertyPath: 'name',
        shapeDataPath: 'Country',
        shapeSettings: {
            colorValuePath: 'Membership',
                colorMapping: [
                {
                    value: 'Permanent', color: '#D84444'
                },
                {
                    value: 'Non-Permanent', color: '#316DB5'
                }]
        }
    }
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs37" %}

Enable Legend

You can show legend for the Maps by setting true to the visible property in legendSettings object and by injecting the Legend

module using provide option.

~/SRC/APP.VUE

```
<template>
    <div id="map">
```

```
        <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import { MapsComponent, LayerDirective, LayersDirective, Legend } from
'@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
export default {
components: {
    'ejs-maps': MapsComponent,
    'e-layer': LayerDirective,
    'e-layers': LayersDirective
},
data () {
    return{
        shapeData: world_map,
        dataSource: [{  "Country": "China", "Membership": "Permanent"},
            {"Country": "France","Membership": "Permanent" },
            { "Country": "Russia","Membership": "Permanent"},
            {"Country": "Kazakhstan","Membership": "Non-Permanent"},
            { "Country": "Poland","Membership": "Non-Permanent"},
            {"Country": "Sweden","Membership": "Non-Permanent"}],
        shapePropertyPath: 'name',
        shapeDataPath: 'Country',
        shapeSettings: {
            colorValuePath: 'Membership',
                colorMapping: [
                {
                    value: 'Permanent', color: '#D84444'
                },
                {
                    value: 'Non-Permanent', color: '#316DB5'
                }]
        },
        legendSettings: {
            visible: true
        }
    }
},
provide: {
    maps: [Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs38" %}

Add Data Label

You can add data labels to show additional information of the shapes in map. This can be achieved by setting visible property to true in the dataLabelSettings object and by injecting DataLabel module using provide option.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings' :dataLabelSettings='dataLabelSettings'></e-
layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import { MapsComponent, LayerDirective, LayersDirective, Legend, DataLabel }
from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
export default {
components: {
    'ejs-maps': MapsComponent,
    'e-layer': LayerDirective,
    'e-layers': LayersDirective
},
data () {
    return{
        shapeData: world_map,
        shapeSettings:{
            autofill: true
        },
        dataLabelSettings: {
            visible: true,
                labelPath: 'name',
                smartLabelMode: 'Trim'
        },
        legendSettings: {
            visible: true
        }
    }
},
provide: {
    maps: [Legend, DataLabel]
}
}
</script>
<style>
```

```
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs39" %}

Enable Tooltip

The tooltip is useful when you cannot display information by using the data labels due to space constraints.

You can enable tooltip by setting the visible property as true

in tooltipSettings object and by injecting `MapsTooltip` module using

`provide` option.

~/SRC/APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps  :titleSettings='titleSettings'
:legendSettings='legendSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings'
:tooltipSettings='tooltipSettings'
:dataLabelSettings='dataLabelSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import { MapsComponent, LayerDirective, LayersDirective, Legend, DataLabel,
MapsTooltip } from '@syncfusion/ej2-vue-maps';
import { usMap } from './usa.js';
import { electionData } from './election-data.js';
export default {
components: {
    'ejs-maps': MapsComponent,
    'e-layer': LayerDirective,
    'e-layers': LayersDirective
},
data () {
    return{
        titleSettings: {
            text: 'USA Election Results - 2016'
        },
        shapeData: usMap,
        dataSource: electionData,
        shapePropertyPath: 'name',
        shapeDataPath: 'State',
        shapeSettings: {
```

```
            colorValuePath: 'Candidate',
            colorMapping: [
            {
                value: 'Trump', color: '#D84444'
            },
            {
                value: 'Clinton', color: '#316DB5'
            }]
        },
        dataLabelSettings: {
            visible: true,
            labelPath: 'State',
            smartLabelMode: 'Trim'
        },
        tooltipSettings: {
            visible: true,
            valuePath: 'State'
        },
        legendSettings: {
            visible: true
        }
    }
},
provide: {
    maps: [Legend, DataLabel, MapsTooltip]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs40" %}

Getting Started with the Vue Maps Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Maps component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

Copyright © 2001 -2024 Syncfusion Inc.

`bash
npm create vite@latest
`

or

`bash
yarn create vite
`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as `my-project` for this article.

`bash
? Project name: » my-project
`

2.Select `Vue` as the framework. It will create a Vue 3 project.

`bash
? Select a framework: » - Use arrow-keys. Return to submit.
Vanilla
Vue
React
Preact
Lit
Svelte
Others
`

3.Choose `JavaScript` as the framework variant to build this Vite project using JavaScript and Vue.

`bash
? Select a variant: » - Use arrow-keys. Return to submit.
JavaScript
TypeScript
Customize with create-vue ↗
Nuxt ↗
`

Copyright © 2001 -2024 Syncfusion Inc.

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Maps component as an example. To use the Vue Maps component in the project, the @syncfusion/ej2-vue-maps package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-maps --save

`

or

`bash

yarn add @syncfusion/ej2-vue-maps

`

Add Syncfusion Vue component

Follow the below steps to add the Vue Maps component using Composition API or Options API:

1.First, import and register the Maps component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { MapsComponent as EjsMaps, LayersDirective as ELayers,
LayerDirective as ELayer, MapAjax , Legend , DataLabel , MapsTooltip } from
'@syncfusion/ej2-vue-maps';
import { Maps } from '@syncfusion/ej2-maps';
Maps.Inject(Legend, DataLabel, MapsTooltip);
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { MapsComponent, LayersDirective, LayerDirective, MapAjax, Legend,
MapAjax, DataLabel, MapsTooltip } from '@syncfusion/ej2-vue-maps'
//Component registration
export default {
name: "App",
components: {
'ejs-maps' : MapsComponent,
'e-layers' : LayersDirective,
'e-layer' : LayerDirective
}
}
</script>
```

2.In the template section, define the Maps component with the layer, title, legend and other settings definitions.

~/SRC/APP.VUE

```
<template>
<ejs-maps :titleSettings='titleSettings' :legendSettings='legendSettings'>
<e-layers>
<e-layer :shapeData='shapeData' :shapePropertyPath='shapePropertyPath'
:shapeDataPath='shapeDataPath' :dataSource='dataSource'
:shapeSettings='shapeSettings' :dataLabelSettings='dataLabelSettings'
:tooltipSettings='tooltipSettings'></e-layer>
</e-layers>
</ejs-maps>
</template>
```

3.In the script section, declare the values for the properties defined in the template section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const titleSettings =  {
text: 'UN security council countries'
};
const shapeData = new MapAjax('https://cdn.syncfusion.com/maps/map-
data/world-map.json');
const dataSource =  [{  "Country": "China", "Membership": "Permanent"},
{"Country": "France","Membership": "Permanent" },
{ "Country": "Russia","Membership": "Permanent"},
{"Country": "Kazakhstan","Membership": "Non-Permanent"},
{ "Country": "Poland","Membership": "Non-Permanent"},
{"Country": "Sweden","Membership": "Non-Permanent"}];
const shapePropertyPath = 'name';
const shapeDataPath = 'Country';
const shapeSettings = {
colorValuePath: 'Membership',
colorMapping: [
{
```

Copyright © 2001 -2024 Syncfusion Inc.

```
value: 'Permanent', color: '#D84444'
},
{
value: 'Non-Permanent', color: '#316DB5'
}
]
};
const dataLabelSettings = {
visible: true,
labelPath: 'name',
smartLabelMode: 'Trim'
};
const legendSettings = {
visible: true
};
const tooltipSettings = {
visible: true,
valuePath: 'Country'
};
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
titleSettings: {
text: 'UN security council countries'
},
shapeData: new MapAjax('https://cdn.syncfusion.com/maps/map-data/world-
map.json'),
dataSource: [{  "Country": "China", "Membership": "Permanent"},
{"Country": "France","Membership": "Permanent" },
{ "Country": "Russia","Membership": "Permanent"},
{"Country": "Kazakhstan","Membership": "Non-Permanent"},
{ "Country": "Poland","Membership": "Non-Permanent"},
{"Country": "Sweden","Membership": "Non-Permanent"}],
shapePropertyPath: 'name',
shapeDataPath: 'Country',
shapeSettings: {
colorValuePath: 'Membership',
colorMapping: [
{
value: 'Permanent', color: '#D84444'
},
{
value: 'Non-Permanent', color: '#316DB5'
}
]
},
dataLabelSettings: {
visible: true,
labelPath: 'name',
smartLabelMode: 'Trim'
},
legendSettings: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
visible: true
},
tooltipSettings: {
visible: true,
valuePath: 'Country'
}
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-maps :titleSettings='titleSettings' :legendSettings='legendSettings'>
<e-layers>
<e-layer :shapeData='shapeData' :shapePropertyPath='shapePropertyPath'
:shapeDataPath='shapeDataPath' :dataSource='dataSource'
:shapeSettings='shapeSettings' :dataLabelSettings='dataLabelSettings'
:tooltipSettings='tooltipSettings'></e-layer>
</e-layers>
</ejs-maps>
</template>
<script setup>
import { MapsComponent as EjsMaps, LayersDirective as ELayers,
LayerDirective as ELayer, MapAjax , Legend , DataLabel , MapsTooltip } from
'@syncfusion/ej2-vue-maps';
import { Maps } from '@syncfusion/ej2-maps';
Maps.Inject(Legend, DataLabel, MapsTooltip);
const titleSettings =  {
text: 'UN security council countries'
};
const shapeData = new MapAjax('https://cdn.syncfusion.com/maps/map-
data/world-map.json');
const dataSource =  [{  "Country": "China", "Membership": "Permanent"},
{"Country": "France","Membership": "Permanent" },
{ "Country": "Russia","Membership": "Permanent"},
{"Country": "Kazakhstan","Membership": "Non-Permanent"},
{ "Country": "Poland","Membership": "Non-Permanent"},
{"Country": "Sweden","Membership": "Non-Permanent"}];
const shapePropertyPath = 'name';
const shapeDataPath = 'Country';
const shapeSettings = {
colorValuePath: 'Membership',
colorMapping: [
{
value: 'Permanent', color: '#D84444'
},
{
value: 'Non-Permanent', color: '#316DB5'
}
]
};
const dataLabelSettings = {
visible: true,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
labelPath: 'name',
smartLabelMode: 'Trim'
};
const legendSettings = {
visible: true
};
const tooltipSettings = {
visible: true,
valuePath: 'Country'
};
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-maps :titleSettings='titleSettings' :legendSettings='legendSettings'>
<e-layers>
<e-layer :shapeData='shapeData' :shapePropertyPath='shapePropertyPath'
:shapeDataPath='shapeDataPath' :dataSource='dataSource'
:shapeSettings='shapeSettings' :dataLabelSettings='dataLabelSettings'
:tooltipSettings='tooltipSettings'></e-layer>
</e-layers>
</ejs-maps>
</template>
<script>
import { MapsComponent, LayersDirective, LayerDirective, MapAjax, Legend,
DataLabel, MapsTooltip } from '@syncfusion/ej2-vue-maps';
// Component registration
export default {
name: "App",
// Declaring component and its directives
components: {
'ejs-maps' : MapsComponent,
'e-layers' : LayersDirective,
'e-layer' : LayerDirective
},
// Bound properties declarations
data() {
return {
titleSettings: {
text: 'UN security council countries'
},
shapeData: new MapAjax('https://cdn.syncfusion.com/maps/map-data/world-
map.json'),
dataSource: [{  "Country": "China", "Membership": "Permanent"},
{"Country": "France","Membership": "Permanent" },
{ "Country": "Russia","Membership": "Permanent"},
{"Country": "Kazakhstan","Membership": "Non-Permanent"},
{ "Country": "Poland","Membership": "Non-Permanent"},
{"Country": "Sweden","Membership": "Non-Permanent"}],
shapePropertyPath: 'name',
shapeDataPath: 'Country',
shapeSettings: {
colorValuePath: 'Membership',
colorMapping: [
{
```

Copyright © 2001 -2024 Syncfusion Inc.

```
value: 'Permanent', color: '#D84444'
},
{
value: 'Non-Permanent', color: '#316DB5'
}
]
},
dataLabelSettings: {
visible: true,
labelPath: 'name',
smartLabelMode: 'Trim'
},
legendSettings: {
visible: true
},
tooltipSettings: {
visible: true,
valuePath: 'Country'
}
};
},
provide: {
maps:[Legend, DataLabel, MapsTooltip]
}
};
</script>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

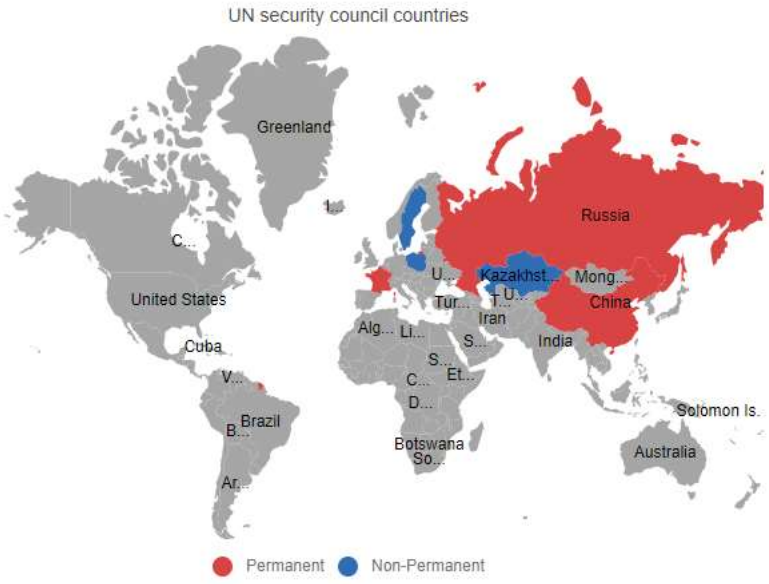
`bash

yarn run dev

`

The output will appear as follows:

Copyright © 2001 -2024 Syncfusion Inc.



Sample: vue3-maps-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Populate data in Vue Maps component

Geometry types

GeoJSON data contains geometry objects with properties such as geometry types and coordinates. The geometry types are the values present in the geometry objects of the GeoJSON data that specify the type of shape to be rendered, as well as the coordinates that help to draw the shape's boundary line. The supportive geometry types are:

| **Shapes** | **Supported** |

| --- | --- |

|Polygon| Yes |

|MultiPolygon| Yes |

|LineString| Yes |

|MultiLineString| Yes |

|Point| Yes |

|MultiPoint| Yes |

|GeometryCollection| Yes |

Copyright © 2001 -2024 Syncfusion Inc.

Shape data

The shape data collection describes geographical shape information that is available in GeoJSON format. The Map shapes are rendered with this data. The custom shapes such as seat selection in bus, seat selection in a cricket stadium and more useful information can be also added as shapeData in the layer of the Maps.

```ts
export let worldMap = // Paste all the content copied from the JSON file.
```

Data source

The dataSource property is used to represent statistical data in the Maps component, and it accepts a collection of values as input. For example, a list of objects as input can be provided to the data source. This data source will be used to color the map, display data labels, and display tooltip, among other things.

The data source is populated with JSON data relative to shape data and stored as JSON object. In the below example, **populationData** can be used as data source in Maps.

```ts
export let populationData: object[] = [
{
'code': 'AF',
'value': 53,
'name': 'Afghanistan',
'population': 29863010,
'density': 119
},
{
'code': 'AL',
'value': 117,
'name': 'Albania',
'population': 3195000,
'density': 111
},
{
'code': 'DZ',
'value': 15,
'name': 'Algeria',
'population': 34895000,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
'density': 15
},
{
'code': 'AO',
'value': 15,
'name': 'Angola',
'population': 18498000,
'density': 15
},
{
'code': 'AR',
'value': 15,
'name': 'Argentina',
'population': 40091359,
'density': 14
},
{
'code': 'AM',
'value': 109,
'name': 'Armenia',
'population': 3230100,
'density': 108
}
];
`
```

Data binding

The following properties in the layers are used for binding data in the Maps component. Both the properties are related to each other.

- shapePropertyPath
- shapeDataPath

shapePropertyPath

The shapePropertyPath property is used to refer the field name in the shapeData property of shape layers to identify the shape. When the values of shapeDataPath property from the dataSource property

and shapePropertyPath property from the shapeData property match, then the associated object from the data source is bound to the corresponding shape.

world-map.js file contains following data and its field **name** value is used to map the corresponding shape with the provided data source.

`javascript

export var world_map = {

"type": "Feature",

"properties": {

"admin": "Afghanistan",

"name": "Afghanistan",

"continent": "Asia"

},

"geometry": { "type": "Polygon", "coordinates": [[[61.21081709172573, https://ej2.syncfusion.com/vue/documentation. },

...

`

shapeDataPath

The shapeDataPath property is similar to the shapePropertyPath property, but it refers to the field name in the dataSource property. For example, populationData contains the **code**, **value**, **name**, **population** and **density** fields. Here, the **name** field is set to the shapeDataPath to map the corresponding value of field name in shape data.

In the below example, both **name** fields contain the same value as **Afghanistan**, this value is matched in both shape data and data source, so that the details associated with **Afghanistan** will be mapped to the corresponding shape and used to color the corresponding shape, display data labels, display tooltips, and more.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map} from './world-map.js';
```

```
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapePropertyPath: 'name',
        shapeDataPath: 'name',
        dataSource: [
            {
                'code': 'AF',
                'value': 53,
                'name': 'Afghanistan',
                'population': 29863010,
                'density': 119,
                'color': '#DEEBAE'
            },
            {
                'code': 'AL',
                'value': 117,
                'name': 'Albania',
                'population': 3195000,
                'density': 111,
                'color': '#A4D6AD'
            },
            {
                'code': 'DZ',
                'value': 15,
                'name': 'Algeria',
                'population': 34895000,
                'density': 15,
                'color': '#37AFAB'
            },
            {
                'code': 'AO',
                'value': 15,
                'name': 'Angola',
                'population': 18498000,
                'density': 15,
                'color': '#547C84'
            },
            {
                'code': 'AR',
                'value': 15,
                'name': 'Argentina',
                'population': 40091359,
                'density': 14,
                'color': '#CEBF93'
            },
            {
                'code': 'AM',
                'value': 109,
                'name': 'Armenia',
                'population': 3230100,
                'density': 108,
                'color': '#a69d70'
            }
        ],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        shapeSettings: {
            colorValuePath: 'color',
            fill: '#E5E5E5'
        }
    }
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs75" %}

Binding complex data source

Data from data source can be bind to the Maps in two different ways.

1. Bind the field name directly to the properties as shapeDataPath, colorValuePath,

valuePath and shapeValuePath.

2. Bind the field name as data.field to the properties as shapeDataPath, colorValuePath,

valuePath and shapeValuePath.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :tooltipSettings='tooltipSettings'
:bubbleSettings='bubbleSettings'
                    :markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsTooltip, Marker, Bubble } from '@syncfusion/ej2-
vue-maps';
import { world_map} from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
```

```
        shapeData: world_map,
        dataSource: [{ "Continent": "North America", 'color': '#71B081',
            data: { "continent": "North America", 'color': '#71B081' }
            },
            { "Continent": "South America", 'color': '#5A9A77',
            data: { "continent": "South America", 'color': '#5A9A77' }
            },
            { "Continent": "Africa", 'color': '#498770',
            data: { "continent": "Africa", 'color': '#498770' }
            },
            { "Continent": "Europe", 'color': '#39776C',
            data: { "continent": "Europe", 'color': '#39776C' }
            },
            { "Continent": "Asia", 'color': '#266665',
            data: { "continent": "Asia", 'color': '#266665' }
            },
            { "Continent": "Australia", 'color': '#124F5E',
            data: { "continent": "Australia", 'color': '#124F5E' }
            }],
        shapePropertyPath: 'continent',
        shapeDataPath: 'data.continent',
        shapeSettings: {
            colorValuePath: 'data.color',
        },
        tooltipSettings: {
            visible: true,
            valuePath: 'data.continent'
        },
         bubbleSettings: [
            {
                visible: true,
                valuePath: 'data.value',
                colorValuePath: 'data.color',
                animationDuration: 0,
                minRadius: 20,
                maxRadius: 90,
                opacity: 0.8,
                dataSource: [
                    { 'name': 'India', 'value': 18.89685398845257,
'population': 391292635,
                    data: { 'color': 'red', 'population': 391292635,
'value': 189685398845257 }
                    }
                ],
                tooltipSettings: {
                    visible: true,
                    valuePath: 'data.population',
                    template:"<div>${data.population}</div>"
                }
            }
        ],
        markerSettings: [
            {
                visible: true,
                dataSource: [
                    { latitude: 37.6276571, longitude: -122.4276688, name:
'San Bruno',
```

```
                    data: { x: 37.6276571, y: -122.4276688, name: 'San
Bruno', shape: 'Pentagon',
                    color: 'red', imageUrl: 'images/ballon.png' }
                    },
                    { latitude: 33.5302186, longitude: -117.7418381, name:
'Laguna Niguel',
                    data: { x: 33.5302186, y: -117.7418381, name: 'Laguna
Niguel', color: 'blue',
                    shape: 'Pentagon', imageUrl: 'images/ballon.png' }
                    }
                ],
                shapeValuePath: "data.shape",
                colorValuePath: "data.color",
                height: 20,
                width: 20,
                offset: {
                    y: -10,
                    x: 0
                },
                longitudeValuePath: "data.y",
                latitudeValuePath: "data.x",
                tooltipSettings: {
                    visible: true,
                    valuePath: 'data.name',
                    format: "${data.name}: ${data.x} : ${data.y}"
                },
                animationDuration: 0
            }
        ]
    }
},
provide: {
    maps: [Marker, Bubble, MapsTooltip]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs76" %}

Layers in Vue Maps component

The Maps component is rendered through layers and any number of layers can be added to the Maps.

Multilayer

The Multilayer support allows loading multiple shape files and map providers in a single container, enabling Maps to display more information. The shape layer or map providers are the main layers of the Maps. Multiple layers can be added as **SubLayer** over the main layers using the type property of layers.

Copyright © 2001 -2024 Syncfusion Inc.

Sublayer

Sublayer is a type of shape file layer. It allows loading multiple shape files in a single map view. For example, a sublayer can be added over the main layer to view geographic features such as rivers, valleys and cities in a map of a country. Similar to the main layer, elements in the Maps such as markers, bubbles, color mapping and legends can be added to the sub-layer.

In this example, the United States map shape is used as shape data by utilizing **usa.ts** file, and **texas.ts** and **california.ts** files are used as sub-layers in the United States map.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings' ></e-layer>
                     <e-layer :shapeData='shapeData1' :type = 'type'
:shapeSettings='shapeSettings1' ></e-layer>
                     <e-layer :shapeData='shapeData2' :type = 'type'
:shapeSettings='shapeSettings2' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { usMap } from './usa.js';
import { california } from './california.js';
import { texas } from './texas.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: usMap,
        shapeSettings: {
            fill: '#E5E5E5',
            border: {
                color: 'black',
                width: 0.1
            }
        },
        shapeData1: texas,
        type: 'SubLayer',
        shapeSettings1: {
            fill: 'rgba(141, 206, 255, 0.6)',
            border: {
                color: '#1a9cff',
                width: 0.25
            }
        },
        shapeData2: california,
        shapeSettings2: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            fill: 'rgba(141, 206, 255, 0.6)',
            border: {
                color: '#1a9cff',
                width: 0.25
            }
        }
    }
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs49" %}

Displaying different layer in the view

Multiple shape files and map providers can be loaded simultaneously in Maps. The baseLayerIndex property is used to determine which layer on the user interface should be displayed. This property is used for the Maps drill-down feature, so when the baseLayerIndex value is changed, the corresponding shape is loaded. In this example, two layers can be loaded with the World map and the United States map. Based on the given baseLayerIndex value the corresponding shape will be loaded in the user interface. If the baseLayerIndex value is set to **0**, then the world map will be loaded.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :baseLayerIndex= "baseLayerIndex">
                <e-layers>
                    <e-layer :shapeData='shapeData'></e-layer>
                    <e-layer :shapeData='shapeData1'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { usMap } from './usa.js';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        baseLayerIndex: 1,
        shapeData: world_map,
        shapeData1: usMap,
    }
},
```

```
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs50" %}

Rendering custom shapes

Custom shapes (also known as custom maps) can be rendered in Maps to represent bus seat booking, cricket stadium, basic home plan/sketch, and so on. To accomplish this, a JSON file in GeoJSON format with proper geometries must be created manually or with the assistance of any online map vendor. The GeoJSON file created must be set to the shapeData in the Maps layer, and the geometryType must be set as **Normal**.

Please refer this link for an example GeoJSON file containing information about bus seat selection.

Please refer this link for more information and a live demonstration.

Providers

Map provider in Vue Maps component

The OpenStreetMap (OSM) is the online Maps provider built by a community of developers; it is free to use under an open license. It allows to view geographical data in a collaborative way from anywhere on the earth. The OSM Maps provides small tile images based on our requests and combines those images into a single image to display the Maps area in the Maps component.

Adding OpenStreetMap

The OSM Maps can be rendered using the urlTemplate property.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :urlTemplate= 'urlTemplate'>
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent } from '@syncfusion/ej2-vue-maps';
Vue.use(MapsPlugin);
export default {
data () {
    return{
       urlTemplate: 'https://tile.openstreetmap.org/level/tileX/tileY.png'
    }
```

```
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs82" %}

Changing the tile server of the OpenStreetMap

The OSM tile server can be changed by setting the tile URL in the urlTemplate property. For more details about the OSM tile server, refer here.

Enabling zooming and panning

The OSM Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a Maps for in-depth analysis. Panning helps to move a Maps around to focus the targeted area.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :zoomSettings= 'zoomSettings'>
                <e-layers>
                    <e-layer :urlTemplate= 'urlTemplate'>
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent, Zoom } from '@syncfusion/ej2-vue-maps';
Vue.use(MapsPlugin);
export default {
data () {
    return{
       zoomSettings: {
           enable: true,
           toolBars: ["Zoom", "ZoomIn", "ZoomOut", "Pan", "Reset"]
       },
       urlTemplate: 'https://tile.openstreetmap.org/level/tileX/tileY.png'
    }
},
provide: {
    maps: [ Zoom ]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
```

```
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs83" %}

Adding markers and navigation line

Markers can be added to the layers of OSM Maps by setting the corresponding location's coordinates of latitude and longitude using e-markerSetting tag. Navigation lines can be added on top of an OSM Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the e-navigationLineSetting tag.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :zoomSettings= 'zoomSettings' :centerPosition=
'centerPosition'>
                <e-layers>
                    <e-layer :urlTemplate= 'urlTemplate'>
                        <e-markerSettings>
                            <e-markerSetting visible= true height=25
width=15 :dataSource ="dataSource" ></e-markerSetting>
                            <e-markerSetting visible= true height=25
width=15 :dataSource ="dataSource1"></e-markerSetting>
                        </e-markerSettings>
                        <e-navigationLineSettings>
                            <e-navigationLineSetting visible = true
:latitude ='latitude' :longitude ='longitude' :color ='color' :angle
='angle' :width="width">
                            </e-navigationLineSetting>
                        </e-navigationLineSettings>
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent, NavigationLine, Marker, Zoom } from
'@syncfusion/ej2-vue-maps';
Vue.use(MapsPlugin);
export default {
data () {
    return{
       zoomSettings: {
           zoomFactor: 4
       },
       centerPosition: {
           latitude: 29.394708,
           longitude: -94.954653
       },
       urlTemplate: 'https://tile.openstreetmap.org/level/tileX/tileY.png',
       dataSource: [
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            {
                latitude: 34.060620,
                longitude: -118.330491,
                name: "California"
        }],
        dataSource1: [
            {
                latitude: 40.724546,
                longitude: -73.850344,
                name: "New York"
        }],
        color: "blue",
        width: 5,
        angle: 0.1,
        latitude: [34.060620, 40.724546],
        longitude: [-118.330491,-73.850344]
    }
},
provide: {
    maps: [ NavigationLine, Marker, Zoom ]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs84" %}

Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the OSM Maps layer for highlighting a particular continent or country in OSM Maps by adding another layer and specifying the type property of Maps layer to "**SubLayer**".

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :urlTemplate= 'urlTemplate'></e-layer>
                    <e-layer :shapeData='shapeData1' :type = 'type'
:shapeSettings='shapeSettings1' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { africa } from './africa.js';
```

```
Vue.use(MapsPlugin);
export default {
data () {
    return{
       urlTemplate: 'https://tile.openstreetmap.org/level/tileX/tileY.png',
       shapeData1: africa,
       type: 'SubLayer',
       shapeSettings1: {
           fill: 'blue'
       }
    }
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs85" %}

Enabling legend

The legend can be added to the tile Maps by setting the visible property of legendSettings to **true**.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps id='container' :legendSettings='legendSettings'>
            <e-layers>
                <e-layer :urlTemplate='urlTemplate'
:shapeSettings='shapeSettings' :markerSettings='markerSettings'></e-layer>
            </e-layers>
        </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Legend } from '@syncfusion/ej2-vue-maps';
import { markerDataSource } from './marker-data.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        urlTemplate: 'https://tile.openstreetmap.org/level/tileX/tileY.png',
        legendSettings: {
            visible: true,
            type: 'Markers',
            useMarkerShape:true,
            toggleLegendSettings: {
                enable: true,
                applyShapeSettings: false,
```

```
                border: {
                    color: 'green',
                    width: 2
                }
            }
        },
        shapeSettings: {
            fill: '#E5E5E5'
        },
        markerSettings: [
            {
                colorValuePath: 'color',
                shapeValuePath:'shape',
                legendText: 'country',
                visible: true,
                dataSource: markerDataSource
            }
        ]
    }
},
provide: {
    maps: [Legend, Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs86" %}

Bing maps in Vue Maps component

Bing Maps is a online Maps provider, owned by Microsoft. As like OSM, it provide Maps tile images based on our requests and combines those images into a single one to display Maps area.

Adding Bing Maps

The Bing Maps can be rendered using the urlTemplate property, which is based on the URL generated by the getBingUrlTemplate method in the Maps. The format of the required URL of Bing Maps varies from other online map providers. As a result, a built-in getBingUrlTemplate method has been included that returns the URL in a generic format. In the meantime, a subscription key is required for Bing Maps. The Bing Maps key can be obtained from here, then append it to the Bing Maps URL before passing it to the getBingUrlTemplate method. The URL returned by this method must be passed to the urlTemplate property.

`ts

<template>

<ejs-maps id='container' :layers='layers' :load='load'>

```
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
Vue.use(MapsPlugin);
export default {
data () {
return {
layers: [
{
}
]
}
},
methods:{
load: function (args) {
let map=document.getElementById('container');
map.ej2_instances[0].getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
map.ej2_instances[0].layers[0].urlTemplate= url;
});
}
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
```

</style>
`

Types of Bing Maps

Bing Maps provides different types of Maps and it is supported in the Maps component.

- **Aerial** - Displays satellite images to highlight roads and major landmarks for easy identification.
- **AerialWithLabel** - Displays aerial Maps with labels for the continent, country, ocean, etc.
- **Road** - Displays the default Maps view of roads, buildings, and geography.
- **CanvasDark** - Displays dark version of the road Maps.
- **CanvasLight** - Displays light version of the road Maps.
- **CanvasGray** - Displays grayscale version of the road Maps.

To render the light version of the road Maps, set the **CanvasLight** value is passed via the URL into the getBingUrlTemplate method demonstrated in the following code sample.

```ts
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps id='container' :centerPosition="centerPosition" :layers='layers' :load='load'>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin} from '@syncfusion/ej2-vue-maps';
Vue.use(MapsPlugin);
export default {
data () {
return{
layers: [{
}],
centerPosition: {
latitude: 38.8951,
longitude: -77.0364
}
}
```

```
},
methods:{
load: function (args) {
let map=document.getElementById('container');
map.ej2_instances[0].getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/CanvasLight?output=json&uriScheme=https&key=?").then(function(url) {
map.ej2_instances[0].layers[0].urlTemplate= url;
});
}
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Enabling zooming and panning

Bing Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a Maps for in-depth analysis. Panning helps to move a Maps around to focus the targeted area.

```
`ts
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps  id='container' :zoomSettings='zoomSettings' :layers='layers' :load='load'>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
```

```
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
enable: true,
toolBars: ["Zoom", "ZoomIn", "ZoomOut", "Pan", "Reset"]
},
layers: [{
}]
}
},
provide: {
maps: [Zoom]
},
methods:{
load: function (args) {
let map=document.getElementById('container');
map.ej2_instances[0].getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
map.ej2_instances[0].layers[0].urlTemplate= url;
});
}
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Copyright © 2001 -2024 Syncfusion Inc.

Adding markers and navigation line

Markers can be added to the layers of Bing Maps by setting the corresponding location's coordinates of latitude and longitude using e-markerSetting tag. Navigation lines can be added on top of an Bing Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the e-navigationLineSetting tag.

```ts
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps id='container' :zoomSettings='zoomSettings' :load='load' :centerPosition= 'centerPosition'>
<e-layers>
<e-layer>
<e-markerSettings>
<e-markerSetting visible= true height=25 width=15 :dataSource ="dataSource" ></e-markerSetting>
<e-markerSetting visible= true height=25 width=15 :dataSource ="dataSource1"></e-markerSetting>
</e-markerSettings>
<e-navigationLineSettings>
<e-navigationLineSetting visible = true :latitude ='latitude' :longitude ='longitude' :color ='color' :angle ='angle' :width="width">
</e-navigationLineSetting>
</e-navigationLineSettings>
</e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom, Marker, NavigationLine } from '@syncfusion/ej2-vue-maps';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
```

```
zoomFactor: 4
},
centerPosition: {
latitude: 29.394708,
longitude: -94.954653
},
dataSource: [
{
latitude: 34.060620,
longitude: -118.330491,
name: "California"
}],
dataSource1: [
{
latitude: 40.724546,
longitude: -73.850344,
name: "New York"
}],
color: "blue",
width: 5,
angle: 0.1,
latitude: [34.060620, 40.724546],
longitude: [-118.330491,-73.850344]
}
},
provide: {
maps: [Zoom, Marker, NavigationLine]
},
methods:{
load: function (args) {
let map=document.getElementById('container');
map.ej2_instances[0].getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
```

```
map.ej2_instances[0].layers[0].urlTemplate= url;
});
}
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the Bing Maps layer for highlighting a particular continent or country in Bing Maps by adding another layer and specifying the type property of Maps layer to **SubLayer**.

```
`
<template>
<div id="app">
<div>
<ejs-maps id='container' :layers='layers' :load='load'>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { africa } from './africa.js';
Vue.use(MapsPlugin);
export default {
data () {
return {
```

```
layers: [{
},
{
layerType: 'Geometry',
type: 'SubLayer',
shapeData: africa,
shapeSettings: {
fill: 'blue'
}
}
]
}
},
methods:{
load: function (args) {
let map=document.getElementById('container');
map.ej2_instances[0].getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
map.ej2_instances[0].layers[0].urlTemplate= url;
});
}
}
}
</script>
```

Enabling legend

The legend can be added to the tile Maps by setting the visible property of legendSettings to **true**.

```
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps id='container' :legendSettings='legendSettings' :load='load'>
<e-layers>
<e-layer :shapeSettings='shapeSettings' :markerSettings='markerSettings'></e-layer>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Legend } from '@syncfusion/ej2-vue-maps';
import { markerDataSource } from './marker-data.js';
Vue.use(MapsPlugin);
export default {
data () {
return {
legendSettings: {
visible: true,
type: 'Markers',
useMarkerShape:true,
toggleLegendSettings: {
enable: true,
applyShapeSettings: false,
border: {
color: 'green',
width: 2
}
}
},
shapeSettings: {
fill: '#E5E5E5'
},
markerSettings: [
{
colorValuePath: 'color',
shapeValuePath:'shape',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
legendText: 'country',
visible: true,
dataSource: markerDataSource
}
]
}
},
provide: {
maps: [Legend, Marker]
},
methods:{
load: function (args) {
let map=document.getElementById('container');
map.ej2_instances[0].getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/Aerial?output=json&uriScheme=https&key=?").then(function(url) {
map.ej2_instances[0].layers[0].urlTemplate= url;
});
}
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Azure maps in Vue Maps component

Azure Maps is yet another online Maps provider, owned by Microsoft. As like OSM and Bing Maps, it provides Maps tile images based on our requests and combines those images into a single one to display Maps area.

Copyright © 2001 -2024 Syncfusion Inc.

Adding Azure Maps

The Azure Maps can be rendered using the urlTemplate property with the tile server URL provided by online map providers. In the meantime, a subscription key is required for Azure Maps. Follow the steps in this link to generate an API key, and then added the key to the URL.

Refer to Azure Maps Licensing.

```ts
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps >
<e-layers>
<e-layer :urlTemplate= 'urlTemplate'>
</e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent } from '@syncfusion/ej2-vue-maps';
Vue.use(MapsPlugin);
export default {
data () {
return{
urlTemplate: 'https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key &api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY'
}
},
}
</script>
<style>
.wrapper {
max-width: 400px;
```

```
margin: 0 auto;
}
</style>
`
```

Enabling zooming and panning

The Azure Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a map for in-depth analysis. Panning helps to move a map around to focus the targeted area.

```ts
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps :zoomSettings= 'zoomSettings'>
<e-layers>
<e-layer :urlTemplate= 'urlTemplate'>
</e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent, Zoom } from '@syncfusion/ej2-vue-maps';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
enable: true,
toolBars: ["Zoom", "ZoomIn", "ZoomOut", "Pan", "Reset"]
},
urlTemplate: 'https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key &api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY'
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
provide: {
maps: [ Zoom ]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Adding markers and navigation line

Markers can be added to the layers of Azure Maps by setting the corresponding location's coordinates of latitude and longitude using e-markerSetting tag. Navigation lines can be added on top of the Azure Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the e-navigationLineSetting tag.

```
`ts
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps :zoomSettings= 'zoomSettings' :centerPosition= 'centerPosition'>
<e-layers>
<e-layer :urlTemplate= 'urlTemplate'>
<e-markerSettings>
<e-markerSetting visible= true height=25 width=15 :dataSource ="dataSource" ></e-markerSetting>
<e-markerSetting visible= true height=25 width=15 :dataSource ="dataSource1"></e-markerSetting>
</e-markerSettings>
<e-navigationLineSettings>
<e-navigationLineSetting visible = true :latitude ='latitude' :longitude ='longitude' :color ='color' :angle ='angle' :width="width">
</e-navigationLineSetting>
</e-navigationLineSettings>
</e-layer>
```

```
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent, NavigationLine, Marker, Zoom } from '@syncfusion/ej2-vue-maps';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
zoomFactor: 4
},
centerPosition: {
latitude: 29.394708,
longitude: -94.954653
},
urlTemplate: 'https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key &api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY',
dataSource: [
{
latitude: 34.060620,
longitude: -118.330491,
name: "California"
}],
dataSource1: [
{
latitude: 40.724546,
longitude: -73.850344,
name: "New York"
}],
```

```
color: "blue",
width: 5,
angle: 0.1,
latitude: [34.060620, 40.724546],
longitude: [-118.330491,-73.850344]
}
},
provide: {
maps: [ NavigationLine, Marker, Zoom ]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the Azure Maps layer for highlighting a particular continent or country in Azure Maps by adding another layer and specifying the type property of Maps layer to **SubLayer**.

```
`
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps >
<e-layers>
<e-layer :urlTemplate= 'urlTemplate'></e-layer>
<e-layer :shapeData='shapeData1' :type = 'type' :shapeSettings='shapeSettings1' ></e-layer>
</e-layers>
</ejs-maps>
</div>
```

```
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent } from '@syncfusion/ej2-vue-maps';
import { africa } from './africa.js';
Vue.use(MapsPlugin);
export default {
data () {
return{
urlTemplate: 'https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key &api-version=1.0&style=satellite&zoom=level&x=tileX&y=tileY',
shapeData1: africa,
type: 'SubLayer',
shapeSettings1: {
fill: 'blue'
}
}
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Enabling legend

The legend can be added to the Azure Maps by setting the visible property of legendSettings to **true**.

```
`
<template>
<div id="app">
<div class='wrapper'>
```

```
<ejs-maps id='container' :legendSettings='legendSettings'>
<e-layers>
<e-layer :urlTemplate='urlTemplate' :shapeSettings='shapeSettings'
:markerSettings='markerSettings'></e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Legend } from '@syncfusion/ej2-vue-maps';
import { markerDataSource } from './marker-data.js';
Vue.use(MapsPlugin);
export default {
data () {
return {
urlTemplate: 'https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key &api-
version=1.0&style=satellite&zoom=level&x=tileX&y=tileY',
legendSettings: {
visible: true,
type: 'Markers',
useMarkerShape:true,
toggleLegendSettings: {
enable: true,
applyShapeSettings: false,
border: {
color: 'green',
width: 2
}
}
},
shapeSettings: {
fill: '#E5E5E5'
```

```
},
markerSettings: [
{
colorValuePath: 'color',
shapeValuePath:'shape',
legendText: 'country',
visible: true,
dataSource: markerDataSource
}
]
}
},
provide: {
maps: [Legend, Marker]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Other maps in Vue Maps component

Apart from OpenStreetMap and Bing Maps, you can also render Maps from other online map service providers by specifying the URL provided by those providers in the urlTemplate property. The URL template concept has been implemented in such a way that any online map service providers using the following template can benefit from previewing their Map in the Syncfusion Vue Maps component.

<!-- markdownlint-disable MD034 -->

Sample Template: https://< domain_name >/maps/basic/{z}/{x}/{y}.png

- "${z}" - It represents zoom factor (level).
- "${x}" - It indicates tile image x-position (tileX).
- "${y}" - It indicates tile image y-position (tileY).

Copyright © 2001 -2024 Syncfusion Inc.

In this case, the key generated for those online map service providers can also be appended to the URL. This allows to create personalized Maps with your own content and imagery.

Following is an example of how to add a TomTom map. You can generate an API key by following the steps in this link and then adding the key to the URL.

```
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps >
<e-layers>
<e-layer :urlTemplate= 'urlTemplate'>
</e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent } from '@syncfusion/ej2-vue-maps';
Vue.use(MapsPlugin);
export default {
data () {
return{
urlTemplate:
"http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key"
}
},
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
```

}

</style>

`

Enabling zooming and panning

Tile Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a Maps for in-depth analysis. Panning helps to move a Maps around to focus the targeted area.

`

<template>

<ejs-maps :zoomSettings= 'zoomSettings'>

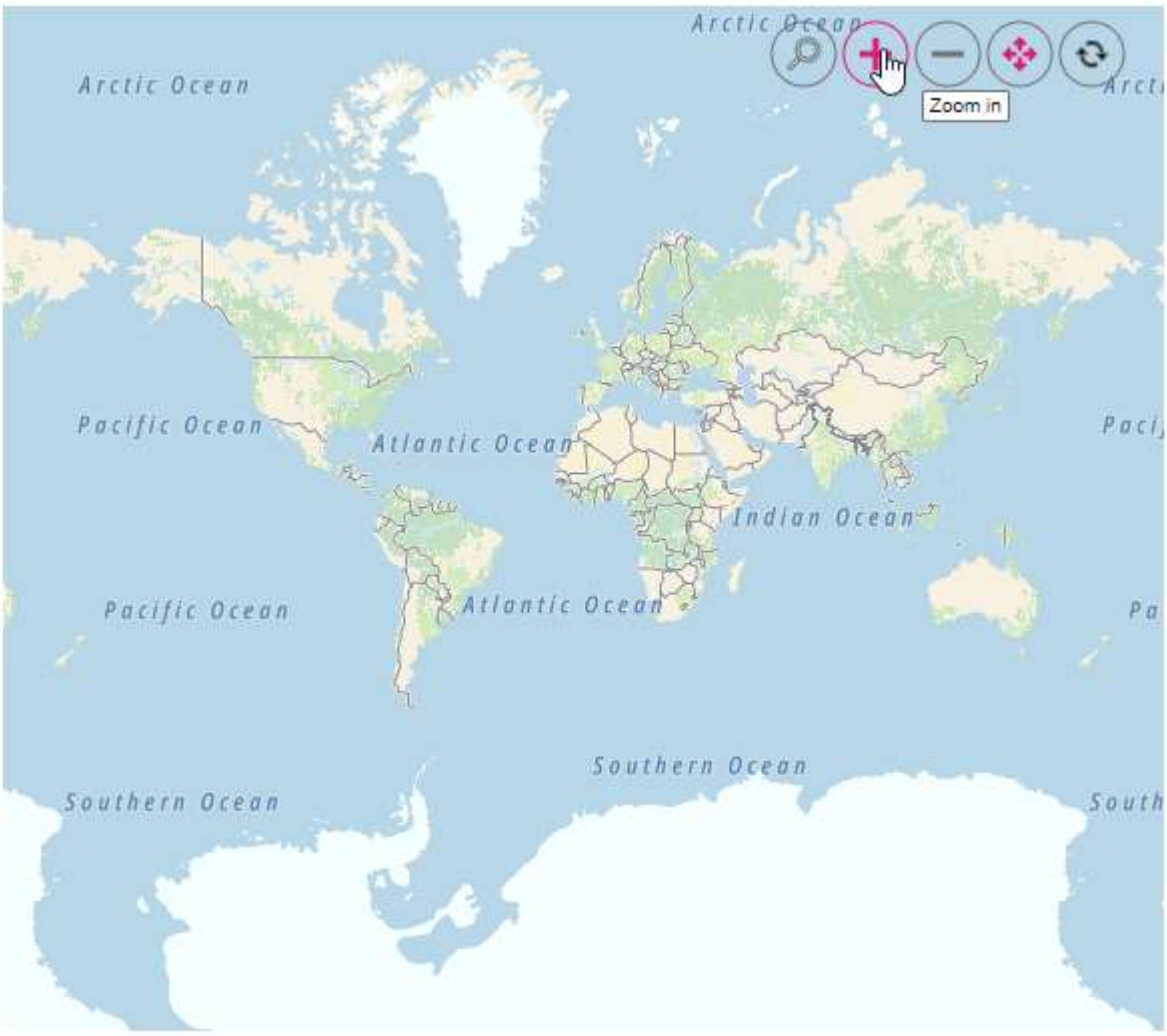
<e-layers>

<e-layer :urlTemplate= "urlTemplate">

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent, Zoom } from '@syncfusion/ej2-vue-maps';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
enable: true,
toolBars: ["Zoom", "ZoomIn", "ZoomOut", "Pan", "Reset"]
},
urlTemplate:
"http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key"
}
},
provide: {
maps: [ Zoom ]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Copyright © 2001 -2024 Syncfusion Inc.



Adding markers and navigation line

Markers can be added to the layers of tile Maps by setting the corresponding location's coordinates of latitude and longitude using e-markerSetting tag. Navigation lines can be added on top of an tile Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the e-navigationLineSetting tag.

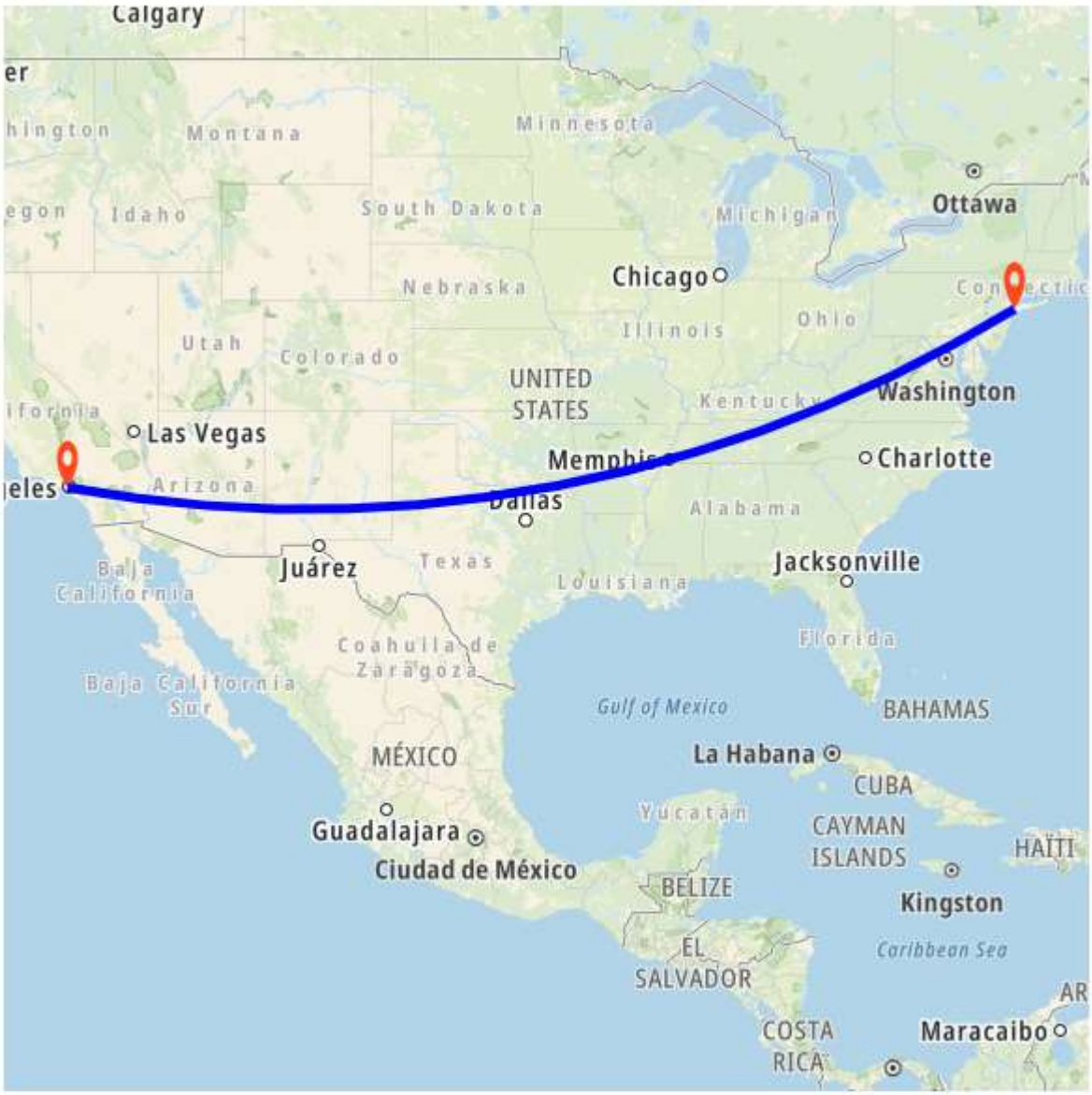
`

```
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps :zoomSettings= 'zoomSettings' :centerPosition="centerPosition">
<e-layers>
<e-layer :urlTemplate= "urlTemplate">
<e-markerSettings>
```

```
<e-markerSetting visible= true height=25 width=15 :dataSource ="dataSource" ></e-markerSetting>
<e-markerSetting visible= true height=25 width=15 :dataSource ="dataSource1"></e-markerSetting>
</e-markerSettings>
<e-navigationLineSettings>
<e-navigationLineSetting visible = true :latitude ='latitude' :longitude ='longitude' :color ='color' :angle ='angle' :width="width">
</e-navigationLineSetting>
</e-navigationLineSettings>
</e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent, NavigationLine, Marker, Zoom } from '@syncfusion/ej2-vue-maps';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
zoomFactor: 4
},
centerPosition: {
latitude: 29.394708,
longitude: -94.954653
},
urlTemplate: "http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key",
dataSource: [
{
latitude: 34.060620,
```

```
longitude: -118.330491,
name: "California"
}],
dataSource1: [
{
latitude: 40.724546,
longitude: -73.850344,
name: "New York"
}],
color: "blue",
width: 5,
angle: 0.1,
latitude: [34.060620, 40.724546],
longitude: [-118.330491,-73.850344]
}
},
provide: {
maps: [ Zoom, NavigationLine, Marker ]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Copyright © 2001 -2024 Syncfusion Inc.



Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the tile Maps layer for highlighting a particular continent or country in tile Maps by adding another layer and specifying the type property of Maps layer to **SubLayer**.

`

```
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps >
<e-layers>
<e-layer :urlTemplate="urlTemplate"></e-layer>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-layer :shapeData='shapeData1' :type = 'type' :shapeSettings='shapeSettings1' ></e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent } from '@syncfusion/ej2-vue-maps';
import { africa } from './africa.js';
Vue.use(MapsPlugin);
export default {
data () {
return{
urlTemplate:
"http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key",
shapeData1: africa,
type: 'SubLayer',
shapeSettings1: {
fill: 'blue'
}
}
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Copyright © 2001 -2024 Syncfusion Inc.

Enabling legend

The legend can be added to the tile Maps by setting the visible property of legendSettings to **true**.

`

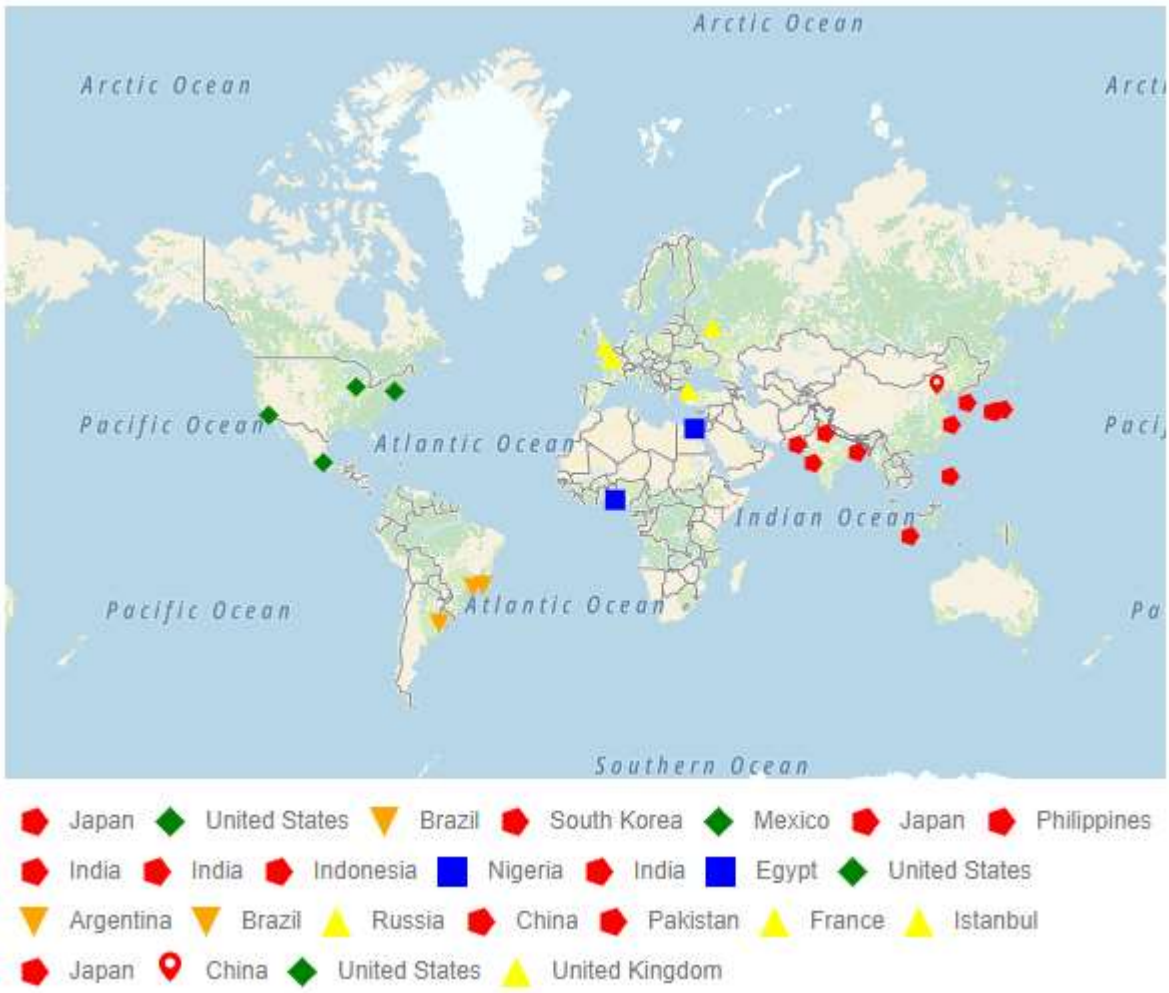
```
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps id='container' :legendSettings='legendSettings'>
<e-layers>
<e-layer :urlTemplate='urlTemplate' :shapeSettings='shapeSettings'
:markerSettings='markerSettings'></e-layer>
</e-layers>
</ejs-maps>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Legend } from '@syncfusion/ej2-vue-maps';
import { markerDataSource } from './marker-data.js';
Vue.use(MapsPlugin);
export default {
data () {
return {
urlTemplate:
'http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key',
legendSettings: {
visible: true,
type: 'Markers',
useMarkerShape:true,
toggleLegendSettings: {
enable: true,
applyShapeSettings: false,
border: {
color: 'green',
width: 2
}
}
},
shapeSettings: {
fill: '#E5E5E5'
},
markerSettings: [
{
colorValuePath: 'color',
shapeValuePath:'shape',
legendText: 'country',
```

```
visible: true,
dataSource: markerDataSource
}
]
}
},
provide: {
maps: [Legend, Marker]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Copyright © 2001 -2024 Syncfusion Inc.



Customization in Vue Maps component

Setting the size for Maps

The width and height of the Maps can be set using the width and height properties in the Maps component. Percentage or pixel values can be used for the height and width values.

APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps height='300px' width='200px'>
                <e-layers>
                    <e-layer :shapeData='shapeData'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
```

```
Vue.use(MapsPlugin);
export default {
    data () {
        return {
            shapeData: world_map
        }
    }
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs19" %}

Maps title

The title for the Maps can be set using the titleSettings. It can be customized using the following properties.

- alignment - To customize the alignment for the text in the title for the Maps. The possible values are **Center**, **Near** and **Far**.
- description - To set the description of the title in Maps.
- text - To set the text for the title in Maps.
- textStyle - To customize the text of the title in Maps.
- subtitleSettings - To customize the subtitle for the Maps.

APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps :titleSettings='titleSettings'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
    data () {
        return {
            titleSettings: {
                text: 'Maps Component',
                textStyle: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    color: 'red',
                    fontStyle: 'italic',
                    fontWeight: 'regular',
                    fontFamily: 'arial',
                    size: '14px'
                },
                alignment: 'Center'
            },
            shapeData: world_map,
            shapeSettings: {
                autofill: true
            }
        }
    }
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs20" %}

Setting theme

The Maps component supports following themes.

- Material
- Fabric
- Bootstrap
- Highcontrast
- MaterialDark
- FabricDark
- BootstrapDark
- Bootstrap4
- HighContrastLight
- Tailwind

By default, the Maps are rendered by the **Material** theme. The theme of the Maps component is changed using the theme property.

APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps theme="MaterialDark">
                <e-layers>
                    <e-layer :shapeData='shapeData'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
```

```
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
    data () {
        return {
            shapeData: world_map
        }
    }
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs21" %}

Customizing Maps container

The following properties are available to customize the container in the Maps.

- background - To apply the background color to the container in the Maps.
- border - To customize the color, width and opacity of the border of the Maps.
- margin - To customize the margins of the Maps.

APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps :background='background' :border='border'
:margin='margin'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
    data () {
        return {
            shapeData: world_map,
            background: '#CCD1D1',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            border: {
                color: 'green',
                width: 2
            },
            margin: {
                bottom: 10,
                left: 20,
                right: 20,
                top: 10
            },
            shapeSettings: {
                autofill: true
            }
        }
    }
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs22" %}

Customizing Maps area

By default, the background color of the shape maps is set as **white**. To modify the background color of the Maps area, the background property in the mapsArea is used. The border of the Maps area can be customized using the border property in the mapsArea.

APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps :mapsArea='mapsArea'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
    data () {
        return {
            shapeData: world_map,
            mapsArea: {
                background: '#CCD1D1',
```

```
                border: {
                    width: 2,
                    color: 'green'
                }
            },
            shapeSettings: {
                autofill: true,
            }
        }
    }
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs23" %}

Customizing the shapes

The following properties are available in shapeSettings to customize the shapes of the Maps.

- fill - To apply the fill color to the all the shapes.
- autofill - To apply the palette colors to the shapes if it is set as true.
- palette - To set the custom palette for the shapes.
- border - To customize the color, width and opacity of the border of the shapes.
- dashArray - To define the pattern of dashes and gaps that is applied to the outline of the shapes.
- opacity - To customize the transparency for the shapes.

APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: world_map,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        shapeSettings: {
            autofill: true,
            palette: ['#C7DE6C', '#59A076', '#88D0BC', '#FEA78C',
'#FFC557'],
            border: {
                color: '#FEE1DD',
                width: 3
            },
            dashArray: '1',
            opacity: 0.9
        }
    }
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs24" %}

Setting color to the shapes from the data source

The color for each shape in the Maps can be set using the colorValuePath property of shapeSettings. The value for the colorValuePath property is the field name from the data source of the shapeSettings which contains the color values.

APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath'
                                :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
    data () {
        return {
            shapeData: world_map,
            shapePropertyPath: 'continent',
            shapeDataPath: 'continent',
```

```
            dataSource: [
                { continent: "North America", color: '#71B081' },
                { continent: "South America", color: '#5A9A77' },
                { continent: "Africa", color: '#498770' },
                { continent: "Europe", color: '#39776C' },
                { continent: "Asia", color: '#266665' },
                { continent: "Oceania", color: '#124F5E' }
            ],
            shapeSettings: {
                colorValuePath: 'color',
            }
        }
    }
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs25" %}

Applying border to individual shapes

The border of each shape in the Maps can be customized using the borderColorValuePath and borderWidthValuePath properties to modify the color and the width of the border respectively. The field name in the data source of the layer which contains the color and the width values must be set in the borderColorValuePath and borderWidthValuePath properties respectively. If the values of borderColorValuePath and borderWidthValuePath do not match with the field name from the data source, then the color and width of the border will be applied to the shapes using the border property in the shapeSettings.

APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath'
                             :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    data () {
        return {
            shapeData: world_map,
            shapePropertyPath: 'continent',
            shapeDataPath: 'continent',
            dataSource: [
                { continent: "North America", color: '#71B081', borderColor:
'#CCFFE5', width: 2 },
                { continent: "South America", color: '#5A9A77', borderColor:
'red', width: 2 },
                { continent: "Africa", color: '#498770', borderColor:
'#FFCC99', width: 2 },
                { continent: "Europe", color: '#39776C', borderColor:
'#66B2FF', width: 2 },
                { continent: "Asia", color: '#266665', borderColor:
'#999900', width: 2 },
                { continent: "Oceania", color: '#124F5E', borderColor:
'blue', width: 2 }
            ],
            shapeSettings: {
                borderColorValuePath: 'borderColor',
                borderWidthValuePath: 'width',
                colorValuePath: 'color'
            }
        }
    }
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs26" %}

Projection type

The Maps component supports the following projection types:

- Mercator
- Equirectangular
- Miller
- Eckert3
- Eckert5
- Eckert6
- Winkel3
- AitOff

By default, the Maps are rendered by the **Mercator** projection type in which the Maps are rendered based on the coordinates. So, the Maps is not stretched. To change the type of projection in the Maps, the projectionType property is used.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="map">
        <div class='wrapper'>
            <ejs-maps projectionType= 'Miller'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: world_map,
        shapeSettings: {
            autofill: true,
            palette: ['#33CCFF', '#FF0000', '#800000', '#FFFF00', '#808000']
        }
    }
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs27" %}

Color mapping in Vue Maps component

Color mapping is used to customize the shape colors based on the given values. It has three types.

1. Range color mapping
2. Equal color mapping
3. Desaturation color mapping.

To add color mapping to the shapes of the Maps, bind the data source to the dataSource property of layerSettings and set the field name which contains the color value in the data source to the colorValuePath property.

Copyright © 2001 -2024 Syncfusion Inc.

Types of color mapping

Range color mapping

Range color mapping applies the color to the shapes of the Maps which matches the numeric values in the data source within the given color mapping ranges. The from and to properties in the colorMapping are used to specify the color mapping ranges in the Maps.

```ts
export let population_density = [
https://ej2.syncfusion.com/vue/documentation.
{
'code': 'AE',
'value': 90,
'name': 'United Arab Emirates',
'population': 8264070,
'density': 99
},
{
'code': 'GB',
'value': 257,
'name': 'United Kingdom',
'population': 62041708,
'density': 255
},
{
'code': 'US',
'value': 34,
'name': 'United States',
'population': 325020000,
'density': 33
}
...
];
```

Bind the **population_density** data to the dataSource property of layerSettings and set the colorValuePath property of shapeSettings as **density**. The range values can be set using the from and to properties of colorMapping.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
import { population_density } from './population-density.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        dataSource: population_density,
        shapeSettings: {
            colorValuePath: 'density',
            fill: '#E5E5E5',
            colorMapping: [
                {
                    from: 0.00001, to: 100, color: 'rgb(153,174,214)'
                },
                {
                    from: 100, to: 200, color: 'rgb(115,143,199)'
                },
                {
                    from: 200, to: 300, color: 'rgb(77,112,184)'
                },
                {
                    from: 300, to: 500, color: 'rgb(38,82,168)'
                },
                {
                    from: 500, to: 19000, color: 'rgb(0,51,153)'
                }
            ]
        }
    }
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs13" %}

Equal color mapping

Equal color mapping applies the color to the shapes of the Maps when the value property of colorMapping matches with the values provided in the data source. The following example shows how to apply equal color mapping to the shapes with the data source **unCountries** which illustrates the permanent and non-permanent countries in the UN security council.

`ts

export let unCountries: object[] = [

{ Country: 'China', Membership: 'Permanent' },

{ Country: 'France', Membership: 'Permanent' },

{ Country: 'Russia', Membership: 'Permanent' },

{ Country: 'United Kingdom', Membership: 'Permanent' },

{ Country: 'United States', Membership: 'Permanent' },

{ Country: 'Bolivia', Membership: 'Non-Permanent' },

{ Country: 'Eq. Guinea', Membership: 'Non-Permanent' },

{ Country: 'Ethiopia', Membership: 'Non-Permanent' },

{ Country: "Côte d'Ivoire", Membership: 'Permanent' },

{ Country: 'Kazakhstan', Membership: 'Non-Permanent' },

{ Country: 'Kuwait', Membership: 'Non-Permanent' },

{ Country: 'Netherlands', Membership: 'Non-Permanent' },

{ Country: 'Peru', Membership: 'Non-Permanent' },

{ Country: 'Poland', Membership: 'Non-Permanent' },

{ Country: 'Sweden', Membership: 'Non-Permanent' },

];

`

Bind the **unCountries** data to the dataSource property of layerSettings and set the colorValuePath property of shapeSettings as **Membership**. Set the value property in the colorMapping to **Permanent** and **Non-Permanent** in the different set of color mapping properties. If the corresponding value of the colorValuePath property matches with the corresponding field name in the data source, then the given color will be applied.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
import { uncountries } from './countries.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeDataPath: 'Country',
        shapePropertyPath: 'name',
        dataSource : uncountries,
        shapeSettings: {
            fill: '#E5E5E5',
            colorMapping: [
                {
                    value: 'Permanent',
                    color: '#EDB46F'
                },
                {
                    color: '#F1931B',
                    value: 'Non-Permanent'
                }
            ],
            colorValuePath: 'Membership'
        }
    }
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs14" %}

Desaturation color mapping

Desaturation color mapping applies the color to the shapes of the Maps, similar to the range color mapping. The opacity will be applied in this color mapping based on the minOpacity and maxOpacity properties in the colorMapping.

Copyright © 2001 -2024 Syncfusion Inc.

The following example shows how to apply desaturation color mapping to the shapes with the data source **population_density** that is available in the Range color mapping section.

Bind the **population_density** data to the dataSource property of layerSettings and set the colorValuePath property of shapeSettings as **density**. The range values can be set using the from and to properties of colorMapping.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
import { population_density } from './population-density.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        dataSource: population_density,
        shapeSettings: {
            colorValuePath: 'density',
            fill: '#E5E5E5',
            colorMapping: [
                {
                    from: 0, to: 100, color: 'red', minOpacity:0.2,
maxOpacity:1
                },
                {
                    from: 101, to: 200, color: 'blue', minOpacity:0.3,
maxOpacity:1
                }
            ]
        }
    }
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
   }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs15" %}

Multiple colors for a single shape

Multiple colors can be added to the color mapping which can be used as gradient effect to a specific shape based on the ranges in the data source. By using the color property of colorMapping, any number of colors can be set to the shapes as a gradient.

The following example demonstrates how to use multiple colors in color mapping with the data source **population_density** that is available in the Range color mapping section.

Bind the **population_density** data to the dataSource property of layerSettings and set the colorValuePath property of shapeSettings as **density**. The range values can be set using the from and to properties of colorMapping.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
import { population_density } from './population-density.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        dataSource: population_density,
        shapeSettings: {
            colorValuePath: 'density',
            fill: '#E5E5E5',
            colorMapping: [
                {
                    from: 0, to: 100, color: ['red','blue']
                },
                {
                    from: 101, to: 200, color: ['green','yellow']
                }
            ]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        }
    }
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs16" %}

Color for items excluded from color mapping

Color mapping can be applied to the shapes in the Maps which does not match color mapping criteria such as range or equal values using the color property of colorMapping.

The following example shows how to set the color for items excluded from the color mapping with the data source **population_density** that is available in the Range color mapping section.

In the following example, color mapping is added for the ranges from 0 to 200. If there are any records in the data source that are outside of this range, the color mapping will not be applied. To apply the color for these excluded items, set the color property alone in the colorMapping.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
import { population_density } from './population-density.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        dataSource: population_density,
        shapeSettings: {
            colorValuePath: 'density',
            fill: '#E5E5E5',
```

```
            colorMapping: [
                {
                    from: 0, to: 100, color: ['red','blue']
                },
                {
                    from: 101, to: 200, color: ['green','yellow']
                },
                {
                    color: 'green'
                }
            ]
        }
    }
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs17" %}

Color mapping for bubbles

The color mapping types such as range color mapping, equal color mapping and desaturation color mapping are applicable for bubbles in the Maps. To add color mapping for bubbles of the Maps, bind the data source to the dataSource property of bubbleSettings and set the field name which contains the color value in the data source to the colorValuePath property. Multiple colors for a single set of bubbles and color for excluded items from colorMapping are also applicable for bubbles.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:bubbleSettings='bubbleSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Bubble } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        bubbleSettings: [{
            visible: true,
            minRadius: 5,
            dataSource: [
                { name: 'India', population: '38332521' },
                { name: 'New Zealand', population: '19651127' },
                { name: 'Pakistan', population: '3090416' }
            ],
            valuePath: 'population',
            colorValuePath: 'population',
            colorMapping: [{
                 value: '38332521',
                 color: '#C3E6ED'
            },
            {
                value: '19651127',
                color: '#F1931B'
            },
            {
                value: '3090416',
                color: 'blue'
            }]
        }]
    }
},
provide: {
    maps: [Bubble]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs18" %}

Data labels in Vue Maps component

Data labels provide information to users about the shapes of the Maps component. It can be enabled by setting the visible property of the dataLabelSettings to **true**.

To use the data label feature, the **DataLabel** module must be injected.

Adding data labels

To display the data labels in the Maps, set the field name containing the text to be displayed from the data source or shape data in the labelPath property of the dataLabelSettings property.

In the following example, the value of labelPath property is set from the field name in the shape data of the Maps layer.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'
                    :dataLabelSettings='dataLabelSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, DataLabel } from '@syncfusion/ej2-vue-maps';
import { usMap } from './usa.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: usMap,
        shapeSettings: {
           autofill:true
        },
        dataLabelSettings: {
            visible: true,
            labelPath: 'name'
        }
    }
},
provide: {
    maps: [DataLabel]
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs28" %}

In the following example, the value of `labelPath` property is set from the field name in the data source of the layer settings.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
```

```
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings'
                    :dataLabelSettings='dataLabelSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, DataLabel } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapePropertyPath: 'name',
        shapeDataPath: 'name',
        dataLabelSettings: {
            visible: true,
            labelPath: "continent",
            smartLabelMode: 'Trim'
        },
        dataSource: [
            {"name": "Afghanistan", "value": 53, "countryCode": "AF",
"population": "29863010", "color": "red", "density": "119", "continent":
"Asia"},
            {"name": "Albania", "value": 117, "countryCode": "AL",
"population": "3195000", "color": "Blue", "density": "111", "continent":
"Europe"},
            {"name": "Algeria", "value": 15, "countryCode": "DZ",
"population": "34895000", "color": "Green", "density": "15", "continent":
"Africa"}
        ],
        shapeSettings: {
            autofill: true
        }
    }
},
provide: {
    maps: [DataLabel]
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs29" %}

Copyright © 2001 -2024 Syncfusion Inc.

Customization

The following properties are available in the dataLabelSettings to customize the data label of the Maps component.

- border - To customize the color, width and opacity for the border of the data labels in Maps.
- fill - To apply the color of the data labels in Maps.
- opacity - To customize the transparency of the data labels in Maps.
- textStyle - To customize the text style of the data labels in Maps.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'
                    :dataLabelSettings='dataLabelSettings'
:tooltipSettings='tooltipSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, DataLabel, MapsTooltip } from '@syncfusion/ej2-vue-
maps';
import { usMap } from './usa.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: usMap,
        shapeSettings: {
           autofill:true
        },
        dataLabelSettings: {
            visible: true,
            smartLabelMode: 'Hide',
            intersectionAction: 'Trim',
            labelPath: 'name',
            border: {
                color: 'green',
                width: 2
            },
            fill: 'transparent',
            opacity: 0.9,
            textStyle: {
                size: '17px',
                fontStyle: 'Sans-serif',
                fontWeight: 'normal'
            }
        },
```

```
        tooltipSettings: {
            visible: true,
            valuePath: 'name'
        }
    }
},
provide: {
    maps: [DataLabel, MapsTooltip]
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs30" %}

Label animation

The data labels can be animated during the initial rendering of the Maps. This can be enabled by setting the animationDuration property in the dataLabelSettings of the Maps. The duration of the animation is specified in milliseconds.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings' :dataLabelSettings='dataLabelSettings'
                        :tooltipSettings='tooltipSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, DataLabel, MapsTooltip } from '@syncfusion/ej2-vue-
maps';
import { usMap } from './usa.js';
Vue.use(MapsPlugin);
export default {
    data() {
        return {
            shapeData: usMap,
            shapeSettings: {
                autofill: true
            },
            dataLabelSettings: {
                visible: true,
                smartLabelMode: 'Hide',
```

```
                intersectionAction: 'Trim',
                labelPath: 'name',
                animationDuration: 4000
            },
            tooltipSettings: {
                visible: true,
                valuePath: 'name'
            }
        }
    },
    provide: {
        maps: [DataLabel, MapsTooltip]
    },
}
</script>
<style>
.wrapper {
    max-width: 400px;
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs106" %}

Smart labels

The Maps component provides an option to handle the labels when they intersect with the corresponding shape borders using the smartLabelMode property. The following options are available in the smartLabelMode property.

- **None** - It specifies that no action is taken, when a label exceeds the shape's region.
- **Hide** - It specifies to hide the labels, when it exceeds the shape's region.
- **Trim** - It specifies to trim the labels, when it exceeds the shape's region.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'
                        :dataLabelSettings='dataLabelSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, DataLabel } from '@syncfusion/ej2-vue-maps';
import { usMap } from './usa.js';
Vue.use(MapsPlugin);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
data () {
    return {
        shapeData: usMap,
        shapeSettings: {
           autofill:true
        },
        dataLabelSettings: {
            visible: true,
            labelPath: 'name',
            smartLabelMode:'Hide'
        }
    }
},
provide: {
    maps: [DataLabel]
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs31" %}

Intersect action

The Maps component provides an option to handle the labels when a label intersects with another label using the intersectionAction property. The following options are available in the intersectionAction property.

- **None** - It specifies that no action is taken, when the labels intersect.
- **Hide** - It specifies to hide the labels when they intersect.
- **Trim** - It specifies to trim the labels when they intersect.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'
                    :dataLabelSettings='dataLabelSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, DataLabel } from '@syncfusion/ej2-vue-maps';
```

```
import { usMap } from './usa.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: usMap,
        shapeSettings: {
           autofill:true
        },
        dataLabelSettings: {
            visible: true,
            labelPath: 'name',
            intersectionAction:'Trim'
        }
    }
},
provide: {
    maps: [DataLabel]
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs32" %}

Adding data label as a template

The custom elements can be added as a template in the data labels by using the template property of dataLabelSettings in the Maps component.

The properties of data label such as, smartLabelMode , intersectionAction, animationDuration, border, fill, opacity and textStyle properties are not applicable to template property. The styles can be applied to the label template using the CSS styles of the HTML element.

APP.VUE

```
<template>
    <div id="template">
        <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings' :shapePropertyPath='shapePropertyPath'
:shapeDataPath='shapeDataPath' :dataSource='dataSource'
                        :dataLabelSettings='dataLabelSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { MapsPlugin, DataLabel } from '@syncfusion/ej2-vue-maps';
import { usMap } from './usa.js';
Vue.use(MapsPlugin);
let contentVue = Vue.component("contentTemplate", {
    template: '<div><div><img
src="https://ej2.syncfusion.com/demos/src/maps/images/weather-clear.png"
style="width:22px;height:22px"> </div> {{data.Name}} </img></div>',
        data() {
            return {
                data: {}
            };
        }
});
let contentTemplate = function() {
    return { template: contentVue };
};
export default {
data () {
    return {
        shapeData: usMap,
        shapeSettings: {
           autofill:true
        },
        shapePropertyPath: 'name',
        shapeDataPath: 'Name',
        dataSource: [
            { "Name": "Iowa", "Population": "29863010" },
            { "Name": "Utah", "Population": "1263010" },
            { "Name": "Texas"," Population": "963010" }
        ],
        dataLabelSettings: {
            visible: true,
            labelPath: 'Name',
            template: contentTemplate
        }
    }
},
provide: {
    maps: [DataLabel]
},
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs33" %}

Polygon shape in Vue Maps component

The Maps component allows you to add polygon shape to a geometry map or an online map by using the properties in the polygons. This section describes how to add polygon shape to the map and customize them.

Copyright © 2001 -2024 Syncfusion Inc.

To render polygon shapes in Maps, **Polygon** module must be injected into the Maps using provide:{ maps:[Polygon] } method. The polygon shape can be rendered over the map layer by defining the points property in the polygons of the Maps component. The points property uses a collection of latitude and longitude values to define the polygon shape.

The polygons provides the following properties for customizing the polygon shape of the Maps component.

- fill - It is used to change the color of the polygon shape.
- opacity - It is used to change the opacity of the polygon shape.
- borderColor - It is used to change the color of the border in the polygon shape.
- borderWidth - It is used to change the width of the border in the polygon shape.
- borderOpacity - It is used to change the opacity of the border in the polygon shape.

You can also include “n” polygon shapes using the polygons property.

The following example shows how to customize the polygon shape over the geometry map.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:polygonSettings='polygonSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Polygon } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
    data() {
        return {
            shapeData: world_map,
            polygonSettings: {
                polygons: [{
                        points: [
                            { longitude: 34.88539587371454, latitude:
28.181421087099537 },
                            { longitude: 37.50029619722466, latitude:
24.299419888989462 },
                            { longitude: 39.22241423764024, latitude:
22.638529461838658 },
                            { longitude: 38.95650769309776, latitude:
21.424998160017495 },
                            { longitude: 40.19963938650778, latitude:
20.271205391339606 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                            { longitude: 41.76547269134551, latitude:
18.315451049867193 },
                            { longitude: 42.78452077838921, latitude:
16.097235052947966 },
                            { longitude: 43.36984949591576, latitude:
17.188572054533054 },
                            { longitude: 44.12558191797012, latitude:
17.407258102232234 },
                            { longitude: 46.69027032797584, latitude:
17.33342243475734 },
                            { longitude: 47.09312386141585, latitude:
16.97087769526452 },
                            { longitude: 48.3417299826302, latitude:
18.152700711188004 },
                            { longitude: 49.74762591400318, latitude:
18.81544363931681 },
                            { longitude: 52.41428026336621, latitude:
18.9035706497573 },
                            { longitude: 55.272683129240335, latitude:
20.133861012918544 },
                            { longitude: 55.60121336079203, latitude:
21.92042703112351 },
                            { longitude: 55.08204399107967, latitude:
22.823302662258882 },
                            { longitude: 52.743894337844154, latitude:
22.954463486477437 },
                            { longitude: 51.47035908651375, latitude:
24.35818837668566 },
                            { longitude: 51.122553219055874, latitude:
24.666679732426346 },
                            { longitude: 51.58731671256831, latitude:
25.173806925822717 },
                            { longitude: 51.35950585992913, latitude:
25.84556484481108 },
                            { longitude: 51.088770529661275, latitude:
26.168494193631147 },
                            { longitude: 50.78527056538036, latitude:
25.349051242147596 },
                            { longitude: 50.88330288802666, latitude:
24.779242606720743 },
                            { longitude: 50.19702490702369, latitude:
25.66825106363693 },
                            { longitude: 50.066461167339924, latitude:
26.268905608606616 },
                            { longitude: 49.645896067213215, latitude:
27.15116474192905 },
                            { longitude: 48.917371072320805, latitude:
27.55738830340198 },
                            { longitude: 48.3984720209192, latitude:
28.566207269716173 },
                            { longitude: 47.68851714518985, latitude:
28.5938991332588 },
                            { longitude: 47.45059089191449, latitude:
29.009321449856984 },
                            { longitude: 44.73549453609391, latitude:
29.157358362696385 },
```

```
                                { longitude: 41.79487372890989, latitude:
31.23489959729713 },
                                { longitude: 40.36977176033773, latitude:
31.9642352513131 },
                                { longitude: 39.168270913149826, latitude:
32.18348471414393 },
                                { longitude: 37.019253492546454, latitude:
31.47710220862595 },
                                { longitude: 37.99644645508337, latitude:
30.4851028633376 },
                                { longitude: 37.67756530485232, latitude:
30.3636358598429 },
                                { longitude: 37.50181466030105, latitude:
29.960155516804974 },
                                { longitude: 36.700288181129594, latitude:
29.882136586478993 },
                                { longitude: 36.100009274845206, latitude:
29.15308642012721 },
                                { longitude: 34.85774476486728, latitude:
29.3103032832622 },
                                { longitude: 34.64498583263142, latitude:
28.135787235699823 },
                                { longitude: 34.88539587371454, latitude:
28.181421087099537 }
                            ],
                            fill: 'red',
                            opacity: 0.7,
                            borderColor: 'green',
                            borderWidth: 2,
                            borderOpacity: 0.7,
                        }]
                }
            }
        },
        provide: {
            maps: [Polygon]
        },
}
</script>
<style>
.wrapper {
    max-width: 400px;
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/polygon-cs1" %}

Markers in Vue Maps component

Markers are notes that are used to leave a message on the Maps. It indicates or marks a specific location with desired symbols on the Maps. It can be enabled by setting the visible property of the markerSettings to **true**.

Copyright © 2001 -2024 Syncfusion Inc.

Adding marker

To add the markers, the dataSource property of the markerSettings has a list of objects that contains the data for markers. Using this property, any number of markers can be added to the layers of the Maps. By default, it displays the markers based on the specified latitude and longitude in the given data source. Each data source object contains the following list of properties.

- latitude - The latitude point which determines the X location of the marker.
- longitude - The longitude point which determines the Y location of the marker.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
      shapeData: world_map,
        markerSettings: [
            {
                dataSource: [
                    { latitude: 49.95121990866204, longitude:
18.468749999999998 },
                    { latitude: 59.88893689676585, longitude: -109.3359375
},
                    { latitude: -6.64607562172573, longitude: -
55.54687499999999 }
                ],
                visible: true,
                height: 20,
                width: 20,
                animationDuration: 0
            }
        ]
    }
},
provide: {
    maps: [Marker]
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs61" %}

Adding marker template

The Marker can be added as a template in the Maps component. The template property of the markerSettings is used to set the HTML string or id of an element as a template.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData' >
                        <e-markerSettings>
                            <e-markerSetting visible= true
:template='contentTemplate' :dataSource ="dataSource" animationDuration = 0
></e-markerSetting>
                            <e-markerSetting visible= true
:template='contentTemplate1' :dataSource ="dataSource1" animationDuration =
0 ></e-markerSetting>
                            <e-markerSetting visible= true
:template='contentTemplate2' :dataSource ="dataSource2" animationDuration =
0 ></e-markerSetting>
                        </e-markerSettings>
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        dataSource: [
          { latitude: 49.95121990866204, longitude: 18.468749999999998 }
       ],
      dataSource1: [
          { latitude: 59.88893689676585, longitude: -109.3359375 }
       ],
        dataSource2: [
          { latitude: -6.646075621725 73, longitude: -55.54687499999999 }
        ],
```

```
        contentTemplate: function () {
            return {
            template: Vue.component('MapsComponent', {
              template: '<div id="marker4" style="color:red"
class="markerTemplate">Europe</div>',
              data() { return {  }; }
            })
          }
        },
        contentTemplate1: function () {
            return {
            template: Vue.component('MapsComponent', {
              template: '<div id="marker5" class="markerTemplate"
style="width:50px;color:blue">NorthAmerica</div>',
              data() { return {  }; }
            })
          }
        },
        contentTemplate2: function () {
            return {
            template: Vue.component('MapsComponent', {
              template: '<div id="marker6" class="markerTemplate"
style="width:50px;color:green">South America </div>',
              data() { return {  }; }
            })
          }
        },
    }
},
provide: {
    maps: [ Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs62" %}

Customization

The following properties are available in markerSettings to customize the Markers of the Maps component.

- border - To customize the color, width and opacity of the border for the markers in Maps.
- fill - To apply the color for markers in Maps.
- dashArray - To define the pattern of dashes and gaps that is applied to the outline of the markers in Maps.
- height - To customize the height of the markers in Maps.
- width - To customize the width of the markers in Maps.
- offset - To customize the position of the markers in Maps.

Copyright © 2001 -2024 Syncfusion Inc.

- opacity - To customize the transparency of the markers in Maps.
- animationDelay - To change the time delay in the transition for markers.
- animationDuration - To change the time duration of animation for markers.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
      shapeData: world_map,
        markerSettings: [{
            visible: true,
            border: {
                color: 'green',
                width: 2
            },
            fill: 'red',
            dashArray: '1',
            height: 20,
            width: 20,
            opacity: 0.9,
            animationDelay: 100,
            animationDuration: 1000,
            shape: 'Balloon',
            dataSource: [
                { latitude: 37.0000, longitude: -120.0000, city:
'California' },
                { latitude: 40.7127, longitude: -74.0059, city: 'New York'
},
                { latitude: 42, longitude: -93, city: 'Iowa' }
            ]
        }]
    }
},
provide: {
    maps: [Marker]
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs63" %}

Marker shapes

The Maps component supports the following marker shapes. To set the shape of the marker, the shape property in markerSettings is used.

- Balloon
- Circle
- Cross
- Diamond
- Image
- Rectangle
- Start
- Triangle
- VerticalLine
- HorizontalLine

Rendering marker shape as image

To render a marker as an image in Maps, set the shape property of markerSettings as **Image** and specify the path of the image to imageUrl property. There is another way to render a marker as an image using the imageUrlValuePath property of the markerSettings. Bind the field name that contains the path of the image in the data source to the imageUrlValuePath property.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
      shapeData: world_map,
```

```
      markerSettings: [{
          visible: true,
          shape: 'Image',
          imageUrl:
'https://ej2.syncfusion.com/vue/demos/src/maps/images/ballon.png',
          height: 20,
          width: 20,
          dataSource: [
              { latitude: 37.0000, longitude: -120.0000, city: 'California'
},
              { latitude: 40.7127, longitude: -74.0059, city: 'New York' },
              { latitude: 42, longitude: -93, city: 'Iowa' }
          ]
      }]
    }
},
provide: {
    maps: [Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs64" %}

Multiple marker groups

Multiple groups of markers can be added in the Maps using the markerSettings in which the properties of markers are added as an array. The customization for the markers can be done with the markerSettings.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    return {
      shapeData: world_map,
      markerSettings: [{
          visible: true,
          shape: 'Diamond',
          height: 15,
          fill: "green",
          width: 15,
          dataSource: [
              { latitude: 37.0000, longitude: -120.0000, name:'California'},
              { latitude: 40.7127, longitude: -74.0059, name:"New York" },
              { latitude: 42, longitude: -93, name:'Iowa' }
          ]
        },
        {
            visible: true,
            dataSource: [
              { latitude: 19.228825, longitude: 72.854118, name: "Mumbai"},
              { latitude: 28.610001, longitude: 77.230003, name: "Delhi" },
              { latitude: 13.067439, longitude: 80.237617, name: "Chennai"}
            ],
            shape: 'Circle',
            fill: "blue",
            height: 10,
            width: 10
        }]
    }
},
provide: {
    maps: [Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs65" %}

Customize marker shapes from data source

Bind different colors and shapes to the marker from data source

Using the shapeValuePath and colorValuePath properties, apply the color and shape of the marker can be applied from the given data source. Bind the data source to the dataSource property of the markerSettings and set the field names that contains the shape and color values in the data source to the shapeValuePath and colorValuePath properties.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
      shapeData: world_map,
        markerSettings: [
            {
                dataSource: [
                  { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'Europe', color:'red', shape:'Triangle' },
                  { latitude: 59.88893689676585, longitude: -109.3359375,
name:'North America',
                  color:'blue', shape:'Pentagon' },
                  { latitude: -6.64607562172573, longitude: -
55.54687499999999, name:'South America',
                color:'green', shape:'InvertedTriangle' }
                ],
                visible: true,
                shapeValuePath:'shape',
                colorValuePath:'color',
            },
        ]
    }
},
provide: {
    maps: [Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs66" %}

Setting value path from the data source

The latitude and longitude values are used to determine the location of each marker in the Maps. The latitudeValuePath and longitudeValuePath properties are used to specify the value path that presents in the data source of the marker. In the following example, the field name from the data source is set to the latitudeValuePath and longitudeValuePath properties.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
      shapeData: world_map,
        markerSettings: [{
            visible: true,
            latitudeValuePath: 'latitude',
            longitudeValuePath: 'longitude',
            dataSource: [
                { latitude: 49.95121990866204, longitude: 18.468749999999998
},
                { latitude: 59.88893689676585, longitude: -109.3359375},
                { latitude: -6.646075621725 73, longitude: -55.54687499999999
}
            ]
         }]
    }
},
provide: {
    maps: [Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs67" %}

Repositioning the marker using drag and drop

The markers on the map can be dragged and dropped to change their position. To enable marker drag and drop, set the enableDrag property to **true** in the markerSettings property.

APP.VUE

```
<template>
    <div id="app">
          <div>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings' :markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, MapsTooltip } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeSettings:{
          fill:'#C3E6ED'
        },
        markerSettings: [{
            visible: true,
            enableDrag:true,
            animationDuration: 0,
            shape: 'Balloon',
            width: 20,
            height: 20,
            border: { width: 2, color: '#285255' },
            tooltipSettings: {
                visible: true,
                valuePath: 'name',
            },
            dataSource: [
                { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'MarkerOne' },
                { latitude: 59.88893689676585, longitude: -109.3359375,
name:'MarkerTwo'},
                { latitude: -6.64607562172573, longitude: -55.54687499999999
, name:'MarkerThree'},
                { latitude: 23.644385824912135, longitude: 77.83189239539234
, name:'MarkerFour'},
                { latitude: 63.66569332894224, longitude: 98.2225173953924 ,
name:'MarkerFive'}
            ]
        }]
    }
},
provide: {
    maps: [Marker, MapsTooltip]
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs100" %}

The data of the drag and dropped marker in the marker data source can be customized using the markerDragStart and markerDragEnd events. When you change the appropriate marker data, the tooltip and legend item text of that marker are automatically updated. The following properties are available in the event argument of the marker drag events.

Argument Name	Description
dataIndex	It represents the index of the data of the dragged marker in the marker data source.
latitude	It represents the latitude coordinate point of the dragged marker.
longitude	It represents the longitude coordinate point for the dragged marker.
markerIndex	It represents the index of the marker setting.
layerIndex	It represents the index of the layer in which the marker belongs.
name	It represents the name of the event.
x	It represents the horizontal location of the mouse pointer on the map when the drag action is performed.
y	It represents the vertical location of the mouse pointer on the map when the drag action is performed.

The following example shows how to use marker drag events to customize the data of the drag and dropped marker in the marker data source.

APP.VUE

```
<template>
    <div id="app">
          <div>
            <ejs-maps ref="maps" :legendSettings='legendSettings'
:markerDragStart='markerDragStart' :markerDragEnd='markerDragEnd'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings' :markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Legend, MapsTooltip} from '@syncfusion/ej2-vue-
maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
data () {
    return{
        shapeData: world_map,
        shapeSettings:{
          fill:'#C3E6ED'
        },
        legendSettings:{
            visible:true,
            type:'Markers',
            shape:'Circle',
            shapeWidth:10,
            shapeHeight:10,
            fill:'#FF471A',
            shapeBorder: { width: 2, color: '#285255' },
        },
        markerSettings: [{
            visible: true,
            enableDrag:true,
            legendText:'name',
            animationDuration: 0,
            shape: 'Balloon',
            width: 20,
            height: 20,
            border: { width: 2, color: '#285255' },
            tooltipSettings: {
                visible: true,
                valuePath: 'name',
            },
            dataSource: [
                { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'MarkerOne' },
                { latitude: 59.88893689676585, longitude: -109.3359375,
name:'MarkerTwo'},
                { latitude: -6.64607562172573, longitude: -55.54687499999999
, name:'MarkerThree'},
                { latitude: 23.644385824912135, longitude: 77.83189239539234
, name:'MarkerFour'},
                { latitude: 63.66569332894224, longitude: 98.2225173953924 ,
name:'MarkerFive'}
            ]
        }]
    }
},
provide: {
    maps: [Marker, Legend, MapsTooltip]
},
methods:{
    markerDragStart: function(args) {
       // When the marker begins to move on the map, the event is triggered.
    },
    markerDragEnd: function(args) {
        // When the marker on the map stops dragging, the event is
triggered.
        let mapsInstance = this.$refs.maps.ej2Instances;
```

```
mapsInstance.layers[args.layerIndex].markerSettings[args.markerIndex].dataSo
urce[args.dataIndex].name = 'Dragged Marker ' + (args.dataIndex + 1);
        mapsInstance.refresh();
    }
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs101" %}

Marker zooming

The Maps can be initially scaled to the center value based on the marker distance. This can be achieved by setting the shouldZoomInitially property in zoomSettings as **true**.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :zoomSettings='zoomSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        zoomSettings: {
          enable: true,
          horizontalAlignment:'Near',
          shouldZoomInitially: true
        },
        shapeData: world_map,
        markerSettings: [
            {
                dataSource:  [
                  { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'Europe' },
                  { latitude: 59.88893689676585, longitude: -109.3359375,
name:'North America' },
                  { latitude: -6.64607562172573, longitude: -
55.54687499999999, name:'South America'}
                ],
                visible: true
            },
        ]
```

```
    }
},
provide: {
    maps: [Marker, Zoom]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs68" %}

Marker Clustering

Maps provide support to cluster the markers when they overlap each other. The number on a cluster indicates how many overlapped markers it contains. If zooming is performed on any of the cluster locations in Maps, the number on the cluster will decrease, and the individual markers will be seen on the map. When zooming out, the overlapping marker will increase. So that it can cluster again and increase the count over the cluster.

To enable clustering in markers, set the allowClustering property of markerClusterSettings as **true** and customization of clustering can be done with the markerClusterSettings.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps  :titleSettings='titleSettings'
:zoomSettings='zoomSettings'  :useGroupingSeparator='useGroupingSeparator'
format='n'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'
:markerClusterSettings='markerClusterSettings'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Zoom, MapsTooltip } from '@syncfusion/ej2-vue-
maps';
import { world_map } from './world-map.js';
import { cluster } from './marker-cluster.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        useGroupingSeparator: true,
        zoomSettings: {
            enable: true,
        },
        shapeData: world_map,
        shapeSettings: {
            fill: '#C1DFF5'
        },
```

```
        titleSettings: {
            text: 'Top 13 largest cities in the World',
            textStyle: {
                size: '16px'
            }
        },
        markerClusterSettings: {
             allowClustering: true,
             shape: 'Circle',
             height: 40,
             width: 40,
             labelStyle : { color: 'white'},
        },
        markerSettings: [
            {
                dataSource: cluster,
                visible: true,
                shape: 'Balloon',
                height:20,
                width:20,
                animationDuration:0,
                tooltipSettings: {
                    visible: true,
                    valuePath: 'area',
                }
            },
        ]
    }
},
provide: {
    maps: [Marker, Zoom, MapsTooltip]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs69" %}

Customization of marker cluster

The following properties are available to customize the marker clustering in the Maps component.

- border - To customize the color, width and opacity of the border of cluster in Maps.
- connectorLineSettings - To customize the connector line in cluster separating the markers.
- dashArray - To customize the dash array for the marker cluster in Maps.
- fill - Applies the color of the cluster in Maps.
- height - To customize the height of the marker cluster in Maps.
- imageUrl - To customize the URL path for the marker cluster when the cluster shape is set as image in Maps.
- labelStyle - To customize the text in marker cluster.

Copyright © 2001 -2024 Syncfusion Inc.

- offset - To customize the offset position for the marker cluster in Maps.
- opacity - To customize the opacity of the marker cluster.
- shape - To customize the shape for the cluster of markers.
- width - To customize the width of the marker cluster in Maps.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps  :titleSettings='titleSettings'
:zoomSettings='zoomSettings'  :useGroupingSeparator='useGroupingSeparator'
format='n'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'
:markerClusterSettings='markerClusterSettings'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Zoom, MapsTooltip, Point } from
'@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
import { cluster } from './marker-cluster.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        useGroupingSeparator: true,
        zoomSettings: {
            enable: true,
        },
        shapeData: world_map,
        shapeSettings: {
            fill: '#C1DFF5'
        },
        titleSettings: {
            text: 'Top 13 largest cities in the World',
            textStyle: {
                size: '16px'
            }
        },
        markerClusterSettings: {
            allowClustering: true,
            allowClusterExpand: true,
            shape: 'Circle',
            height: 40,
            width: 40,
            labelStyle : { color: 'white'},
            offset: new Point(10, 20),
            opacity: 0.9,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            fill: 'green',
            connectorLineSettings: {
                color: 'orange',
                opacity: 0.8,
                width: 2
            }
        },
        markerSettings: [
            {
                dataSource: cluster,
                visible: true,
                shape: 'Balloon',
                height:20,
                width:20,
                animationDuration:0,
                tooltipSettings: {
                    visible: true,
                    valuePath: 'area',
                }
            },
        ]
    }
},
provide: {
    maps: [Marker, Zoom, MapsTooltip]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs70" %}

Expanding the marker cluster

The cluster is formed by grouping an identical and non-identical marker from the surrounding area. By clicking on the cluster and setting the allowClusterExpand property in markerClusterSettings as **true** to expand the identical markers. If zooming is performed in any of the locations of the cluster, the number on the cluster will decrease and the overlapping marker will be split into an individual marker on the map. When performing zoom out, it will increase the marker count and then cluster it again.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps  :zoomSettings='zoomSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerClusterSettings='markerClusterSettings'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
```

```
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        zoomSettings: {
            enable: true,
            mouseWheelZoom : true,
        },
        shapeData: world_map,
        markerClusterSettings: {
            allowClustering: true,
            allowClusterExpand: true,
            shape: 'Circle',
            height: 40,
            width: 40,
            labelStyle : { color: 'white'},
        },
        markerSettings: [
            {
                dataSource:[
                  { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'Europe' },
                  { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'Europe' },
                  { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'Europe' },
                  { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'Europe' },
                  { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'Europe' },
                  { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'Europe' },
                  { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'Europe' },
                  { latitude: 59.88893689676585, longitude: -109.3359375,
name:'North America' },
                  { latitude: -6.64607562172573, longitude: -
55.54687499999999, name:'South America'}
                ] ,
                visible: true
            },
        ]
    }
},
provide: {
    maps: [Marker, Zoom ]
}
}
</script>
<style>
```

```
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs71" %}

Tooltip for marker

Tooltip is used to display more information about a marker on mouse over or touch end event. This can be enabled separately for marker by setting the visible property of tooltipSettings to **true**. The valuePath property in the tooltipSettings takes the field name that presents in dataSource and displays that value as tooltip text.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, MapsTooltip } from '@syncfusion/ej2-vue-maps';
import { usMap } from './usa.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: usMap,
        markerSettings: [{
            dataSource: [
                { latitude: 40.7424509, longitude: -74.0081468, city: 'New
York' }
            ],
        visible:true,
        shape:'Circle',
        fill:'white',
        width:3,
        animationDuration:0,
        border: { width:2, color:'green'},
        tooltipSettings: {
            visible: true,
            valuePath:'city'
        }
        }]
    }
},
provide: {
```

```
    maps: [Marker, MapsTooltip]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs72" %}

Bubble in Vue Maps component

Bubbles in the Maps component represent the underlying data values of the Maps. It can be scattered throughout the Maps shapes that contain values in the data source. Bubbles are enabled by setting the visible property of bubbleSettings to **true**. To add bubbles to the Maps, bind the data source to the dataSource property of bubbleSettings and set the field name, that contains the numerical data, in the data source to the valuePath property.

`ts

export let world_map = // paste the world map from worldMap.json GeoJSON file.

`

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:bubbleSettings='bubbleSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Bubble } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
       bubbleSettings: [{
            visible: true,
            minRadius: 20,
            dataSource: [
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                { name: 'India', population: '38332521' },
                { name: 'New Zealand', population: '19651127' },
                { name: 'Pakistan', population: '3090416' }
            ],
            maxRadius: 40,
            valuePath: 'population'
        }]
    }
},
provide: {
    maps: [Bubble]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs6" %}

Bubble shapes

The following types of shapes are available to render the bubbles in Maps.

- Circle
- Square

By default, bubbles are rendered in the **Circle** type. To change the type of the bubble, set the bubbleType property of bubbleSettings as **Square** to render the square shape bubbles.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:bubbleSettings='bubbleSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Bubble } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
```

```
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        bubbleSettings: [{
            visible: true,
            bubbleType: 'Square',
            dataSource: [
                { name: 'India', population: '38332521' },
                { name: 'Pakistan', population: '3090416' }
            ],
            valuePath: 'population'
        }]
    }
},
provide: {
    maps: [Bubble]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs7" %}

Customization

The following properties are available in bubbleSettings to customize the bubbles of the Maps component.

- border - To customize the color, width and opacity of the border of the bubbles in Maps.
- fill - To apply the color for bubbles in Maps.
- opacity - To apply opacity to the bubbles in Maps.
- animationDelay - To change the time delay in the transition for bubbles.
- animationDuration - To change the time duration of animation for bubbles.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:bubbleSettings='bubbleSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { MapsPlugin, Bubble } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        bubbleSettings: [{
            visible: true,
            minRadius: 5,
            dataSource: [
                { name: 'India', population: '38332521' },
                { name: 'New Zealand', population: '19651127' },
                { name: 'Pakistan', population: '3090416' }
            ],
            fill: 'green',
            animationDelay: 100,
            animationDuration: 1000,
            maxRadius: 20,
            border: {
                color: 'blue',
                width: 2
            },
            opacity: 1,
            valuePath: 'population'
        }]
    }
},
provide: {
    maps: [Bubble]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs8" %}

Setting colors to the bubbles from the data source

The color for each bubble in the Maps can be set using the colorValuePath property of bubbleSettings. The value for the colorValuePath property is the field name from the data source of the bubbleSettings which contains the color values.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:bubbleSettings='bubbleSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Bubble } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        bubbleSettings: [{
            visible: true,
            minRadius: 20,
            colorValuePath: 'color',
            dataSource: [
                { name: 'India', population: '38332521', color: 'blue' },
                { name: 'New Zealand', population: '19651127', color:
'#c2d2d6'  },
                { name: 'Pakistan', population: '3090416', color: '#09156d'
}
            ],
            maxRadius: 40,
            valuePath: 'population'
        }]
    }
},
provide: {
    maps: [Bubble]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs9" %}

Setting the range of the bubble size

The size of the bubbles is calculated from the values got from the valuePath property. The range for the radius of the bubbles can be modified using minRadius and maxRadius properties.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:bubbleSettings='bubbleSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Bubble } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
       bubbleSettings: [{
            visible: true,
            minRadius: 5,
            dataSource: [
                { name: 'India', population: '38332521' },
                { name: 'New Zealand', population: '19651127' },
                { name: 'Pakistan', population: '3090416' }
            ],
            maxRadius: 80,
            valuePath: 'population'
        }]
    }
},
provide: {
    maps: [Bubble]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs10" %}

Multiple bubble groups

Multiple groups of bubbles can be added to the Maps using the bubbleSettings in which the properties of bubbles are added as an array. The customization for the bubbles can be done with the

Copyright © 2001 -2024 Syncfusion Inc.

bubbleSettings. In the following example, the gender-wise population ratio is demonstrated with two different bubble groups.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:bubbleSettings='bubbleSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Bubble } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        bubbleSettings: [{
            visible: true,
            minRadius: 5,
            valuePath: "femaleRatio",
            colorValuePath: "femaleRatioColor",
            dataSource: [
                {
                    country: "United States", femaleRatio: 50.50442726,
maleRatio: 49.49557274, femaleRatioColor: "green", maleRatioColor: "blue"
                },
                {
                    country: "India", femaleRatio: 48.18032713, maleRatio:
51.81967287, femaleRatioColor: "blue", maleRatioColor: "#c2d2d6"
                },
                {
                    country: "Oman", femaleRatio: 34.15597234, maleRatio:
65.84402766, femaleRatioColor: "#09156d", maleRatioColor: "orange"
                },
                {
                    country: "United Arab Emirates", femaleRatio:
27.59638942, maleRatio: 72.40361058, femaleRatioColor: "#09156d",
maleRatioColor: "orange"
                }
            ],
            maxRadius: 20,
        },
        {
            visible: true,
```

```
            bubbleType: 'Circle',
            opacity: 0.4,
            minRadius: 15,
            valuePath: "maleRatio",
            colorValuePath: "maleRatioColor",
            dataSource: [
                {
                    country: "United States", femaleRatio: 50.50442726,
maleRatio: 49.49557274, femaleRatioColor: "green", maleRatioColor: "blue"
                },
                {
                    country: "India", femaleRatio: 48.18032713, maleRatio:
51.81967287, femaleRatioColor: "blue", maleRatioColor: "#c2d2d6"
                },
                {
                    country: "Oman", femaleRatio: 34.15597234, maleRatio:
65.84402766, femaleRatioColor: "#09156d", maleRatioColor: "orange"
                },
                {
                    country: "United Arab Emirates", femaleRatio:
27.59638942, maleRatio: 72.40361058, femaleRatioColor: "#09156d",
maleRatioColor: "orange"
                }
            ],
            maxRadius: 25,
        }]
    }
},
provide: {
    maps: [Bubble]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs11" %}

Enable tooltip for bubble

The tooltip for the bubbles can be enabled by setting the visible property of the tooltipSettings as **true**. The content for the tooltip can be set using the valuePath property in the tooltipSettings of the bubbleSettings where the value for the valuePath property is the field name from the data source of the bubbleSettings. Also added any HTML element as the template in tooltip using the template property.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:bubbleSettings='bubbleSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Bubble, MapsTooltip } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        bubbleSettings: [{
            visible: true,
            minRadius: 5,
            dataSource: [
                { name: 'India', population: '38332521' },
                { name: 'New Zealand', population: '19651127' },
                { name: 'Pakistan', population: '3090416' }
            ],
            maxRadius: 20,
            valuePath: 'population',
            tooltipSettings: {
                visible: true,
                valuePath: 'population',
            }
        }]
    }
},
provide: {
    maps: [Bubble, MapsTooltip]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs12" %}

Legend in Vue Maps component

A Legend is a visual representation of the symbols used on the Maps. It can be represented in various colors, shapes or other identifiers based on the data and provides valuable information for interpreting what the Maps are displaying. It explains what each symbol in the Maps represents. Legends are enabled by setting the visible property of legendSettings to **true**.

Copyright © 2001 -2024 Syncfusion Inc.

Modes of legend

Legend had two types of mode.

1. **Default** mode
2. **Interactive** mode

Default mode

Default mode legends having symbols with legend labels, used to identify the shape or bubble or marker color. To enable this option by setting the mode property of legendSettings as **Default**.

Interactive mode

The legends can be made interactive with an arrow mark indicating the exact range color in the legend when the mouse hovers over the corresponding shapes. To enable this type of mode by setting the mode property of legendSettings as **Interactive**. The invertedPointer property is used to enable or disable the visibility of the inverted pointer in interactive legend in Maps.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Legend } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        legendSettings: {
            visible: true,
            mode: 'Interactive',
            invertedPointer: true
        },
        shapeData: world_map,
        dataSource: [
            { "Country": "China", "Membership": "Permanent" },
            { "Country": "France", "Membership": "Permanent" },
            { "Country": "Russia", "Membership": "Permanent" },
            { "Country": "Kazakhstan", "Membership": "Non-Permanent" },
            { "Country": "Poland", "Membership": "Non-Permanent" },
            { "Country": "Sweden", "Membership": "Non-Permanent" }
        ],
        shapePropertyPath: 'name',
        shapeDataPath: 'Country',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        shapeSettings: {
            colorValuePath: 'Membership',
            colorMapping: [
                {
                    value: 'Permanent', color: '#D84444'
                },
                {
                    value: 'Non-Permanent', color: '#316DB5'
                }
            ]
        }
    }
},
provide: {
    maps: [Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs51" %}

Positioning of the legend

The legend can be positioned in the following two ways:

- Absolute position
- Dock position

Absolute position

The legend of the Maps can be positioned using the location property in the legendSettings. For positioning the legend based on co-ordinates corresponding to a Maps, the position property is set as **Float**.

Dock position

Legends are positioned in the following locations within the container. The position property in legendSettings is used to set these options in Maps.

- Top
- Left
- Bottom
- Right

The above four positions can be aligned with combination of **Near**, **Center** and **Far** using alignment property in legendSettings. So, the legend can be aligned to 12 positions.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Legend } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        legendSettings: {
            visible: true,
            position: 'Top',
            alignment: 'Near'
        },
        shapeData: world_map,
        dataSource: [
            { "Country": "China", "Membership": "Permanent" },
            { "Country": "France", "Membership": "Permanent" },
            { "Country": "Russia", "Membership": "Permanent" },
            { "Country": "Kazakhstan", "Membership": "Non-Permanent" },
            { "Country": "Poland", "Membership": "Non-Permanent" },
            { "Country": "Sweden", "Membership": "Non-Permanent" }
        ],
        shapePropertyPath: 'name',
        shapeDataPath: 'Country',
        shapeSettings: {
            colorValuePath: 'Membership',
            colorMapping: [
                {
                    value: 'Permanent', color: '#D84444'
                },
                {
                    value: 'Non-Permanent', color: '#316DB5'
                }
            ]
        }
    }
},
provide: {
    maps: [Legend]
}
}
</script>
<style>
  .wrapper {
```

```
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs52" %}

Legend for shapes

Legend for shapes can be generated from color mapping types such as equal color mapping, range color mapping and desaturation color mapping.

The below code snippet demonstrate the equal color mapping legends for the shapes. To bind the given data source to the dataSource property of layerSettings. Set the value of shapePropertyPath to **name** and shapeDataPath to **Country**. To enable equal color mapping, set the colorMapping as an array in shapeSettings. Finally, set the visible property of legendSettings as **true**. The label property in colorMapping is used to set the text name for legend in Maps.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings'>
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Legend } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        legendSettings: {
            visible: true
        },
        shapeData: world_map,
        dataSource: [
            {  "Country": "China", "Membership": "Permanent" },
            { "Country": "France","Membership": "Permanent" },
            { "Country": "Russia","Membership": "Permanent" },
            { "Country": "Kazakhstan","Membership": "Non-Permanent" },
            { "Country": "Poland","Membership": "Non-Permanent" },
            { "Country": "Sweden","Membership": "Non-Permanent" }
        ],
        shapePropertyPath: 'name',
        shapeDataPath: 'Country',
        shapeSettings: {
```

```
            colorValuePath: 'Membership',
            colorMapping: [
                { value: 'Permanent', color: '#D84444' },
                { value: 'Non-Permanent', color: '#316DB5' }
            ]
        }
    }
},
provide: {
    maps: [Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs53" %}

Legend shape

Maps supports the following types of legend shapes. The shape property in the legendSettings can be used to change the type of legend shapes.

- Circle
- Rectangle
- Triangle
- Diamond
- Cross
- Star
- HorizontalLine
- VerticalLine
- Pentagon
- InvertedTriangle

The shape of legends can be customized by using the shapeWidth, shapeHeight, shapeBorder and shapePadding properties.

Customization

The following properties are available in legend to customize the legend shape and legend text in Maps.

- background - To customize the background color of the Legend.
- border - To customize the color, width and opacity of the border for the Legend.
- fill - To apply the color for the Legend.
- labelDisplayMode - To customize the display mode for the Legend text.
- labelPosition - To customize the position of the Legend text.
- orientation - To customize the orientation of the Legend.
- textStyle - To customize the text style for Legend.
- title - To apply the title for the Legend.

Copyright © 2001 -2024 Syncfusion Inc.

- titleStyle - To customize the style of the title for the Legend.
- height - To customize the height of the Legend.
- width - To customize the width of the Legend.
- opacity - To apply the opacity to the Legend.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Legend } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        legendSettings: {
            visible: true,
            background: 'green',
            border: {
                color: 'blue',
                width: 2
            },
            fill: 'orange',
            labelPosition: 'Before',
            orientation: 'Vertical',
            textStyle: {
                size: '12px',
                color: 'red',
                fontStyle: 'italic'
            },
            title: {
                description: 'Legend title',
                text: 'Legend'
            },
            titleStyle: {
                size: '12px',
                color: '#d6e341',
                fontStyle: 'italic'
            }
        },
        shapeData: world_map,
        dataSource: [
            { "Country": "China", "Membership": "Permanent" },
```

```
            { "Country": "France", "Membership": "Permanent" },
            { "Country": "Russia", "Membership": "Permanent" },
            { "Country": "Kazakhstan", "Membership": "Non-Permanent" },
            { "Country": "Poland", "Membership": "Non-Permanent" },
            { "Country": "Sweden", "Membership": "Non-Permanent" }
        ],
        shapePropertyPath: 'name',
        shapeDataPath: 'Country',
        shapeSettings: {
            colorValuePath: 'Membership',
            colorMapping: [
                {
                    value: 'Permanent', color: '#D84444'
                },
                {
                    value: 'Non-Permanent', color: '#316DB5'
            }]
        }
    }
},
provide: {
    maps: [Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs54" %}

Legend for items excluded from color mapping

The legend can be enabled for items excluded from the color mapping using the color property in colorMapping. Refer to the **population_density** data which demonstrates the population density of some countries.

`ts

export let population_density = [

https://ej2.syncfusion.com/vue/documentation.

{

'code': 'AE',

'value': 90,

'name': 'United Arab Emirates',

'population': 8264070,

'density': 99

},

Copyright © 2001 -2024 Syncfusion Inc.

{

'code': 'GB',

'value': 257,

'name': 'United Kingdom',

'population': 62041708,

'density': 255

},

{

'code': 'US',

'value': 34,

'name': 'United States',

'population': 325020000,

'density': 33

}

...

];

`

In the following example, color mapping is added for the ranges from **0** to **200**. If there are any records in the data source that are outside of this range, the color mapping will not be applied. To apply the color for these excluded items, set the color property alone in the colorMapping. To enable legend for these items, set the visible property of legendSettings to **true**.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Legend } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
import { population_density } from './population-density.js';
Vue.use(MapsPlugin);
export default {
```

```
data () {
    return {
        shapeData: world_map,
        legendSettings: {
            visible: true
        },
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        dataSource: population_density,
        shapeSettings: {
            colorValuePath: 'density',
            colorMapping: [
                {
                    from: 0, to: 100, color: ['red','blue']
                },
                {
                    from: 101, to: 200, color: ['green','yellow']
                },
                {
                    color: 'green'
                }
            ]
        }
    }
},
provide: {
    maps: [Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs55" %}

Hide desired legend items

Use the showLegend property in the colorMapping to show or hide the desired legend items in Maps. If the showLegend property is set to **false**, the legend item will be hidden. otherwise, it will be visible.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Legend } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
import { population_density } from './population-density.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: world_map,
        legendSettings: {
            visible: true
        },
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        dataSource: population_density,
        shapeSettings: {
            colorValuePath: 'density',
            colorMapping: [
                {
                    from: 0, to: 100, color: 'rgb(153,174,214)', showLegend:
true
                },
                {
                    from: 101, to: 200, color: 'rgb(115,143,199)',
showLegend: false
                },
                {
                    color: 'rgb(77,112,184)', showLegend: false
                }
            ]
        }
    }
},
provide: {
    maps: [Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs56" %}

Hide legend items based on data source value

Depending on the boolean values provided in the data source, the legend items will be hidden or visible. Bind the field name that contains the visibility state in the data source to the showLegendPath property of the legendSettings to achieve this.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Legend } from '@syncfusion/ej2-vue-maps';
import { World_Map } from './worldmap.js';
import { default_data } from './legenddata.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: World_Map,
        legendSettings: {
            visible: true,
            showLegendPath:'visibility'
        },
        shapeDataPath: 'continent',
        shapePropertyPath: 'continent',
        dataSource: default_data,
        shapeSettings: {
            colorValuePath: 'color',
        }
    }
},
provide: {
    maps: [Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/color-mapping-cs1" %}

Binding legend item text from data source

To show the legend text based on values provided in the data source, use the valuePath property in the legendSettings.

APP.VUE

```
<template>
```

```
    <div id="app">
        <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Legend } from '@syncfusion/ej2-vue-maps';
import { World_Map } from './worldmap.js';
import { default_data } from './legenddata.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: World_Map,
        legendSettings: {
            visible: true,
            valuePath:'continent'
        },
        shapeDataPath: 'continent',
        shapePropertyPath: 'continent',
        dataSource: default_data,
        shapeSettings: {
            colorValuePath: 'color',
        }
    }
},
provide: {
    maps: [Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/color-mapping-cs2" %}

Hide duplicate legend items

To hide the duplicate legend items in Maps, set the removeDuplicateLegend property to **true** in the legendSettings.

APP.VUE

```
<template>
    <div id="app">
```

```
        <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Legend } from '@syncfusion/ej2-vue-maps';
import { World_Map } from './worldmap.js';
import { default_data } from './legenddata.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: World_Map,
        legendSettings: {
            visible: true,
            valuePath: 'continent',
            removeDuplicateLegend: true
        },
        shapeDataPath: 'continent',
        shapePropertyPath: 'continent',
        dataSource: default_data,
        shapeSettings: {
            colorValuePath: 'color',
        }
    }
},
provide: {
    maps: [Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/color-mapping-cs3" %}

Toggle option in legend

The toggle option has been provided for legend. If the legend can be toggled, the given color will be changed to the corresponding Maps shape item. To enable the toggle options in Legend, set the enable property of the toggleLegendSettings to **true**.

The following properties are available to customize the toggle option in legend.

Copyright © 2001 -2024 Syncfusion Inc.

- applyShapeSettings – To apply the fill property value to the shape of the Maps when toggling the legend items.
- fill - To apply the color to the shape of the Maps for which legend item is toggled.
- opacity – To customize the transparency for the shapes for which legend item is toggled.
- border – To customize the color, width and opacity of the border of the shapes in Maps.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Legend } from '@syncfusion/ej2-vue-maps';
import { World_Map } from './worldmap.js';
import { Population_Density } from './populationdensity.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: World_Map,
        legendSettings: {
            visible: true,
            toggleLegendSettings: {
                enable: true,
                applyShapeSettings: false,
                border: {
                    color: "green",
                    width: 2
                }
            }
        },
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        dataSource: Population_Density,
        shapeSettings: {
            colorValuePath: 'density',
             colorMapping: [
                {
                    from: 0, to: 100, color: 'rgb(153,174,214)',
                },
                {
                    from: 101, to: 200, color: 'rgb(115,143,199)',
                },
                {
                    color: 'rgb(77,112,184)'
```

```
                },
            ]
        }
    }
},
provide: {
    maps: [Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/color-mapping-cs4" %}

Enable legend for bubbles

To enable the legend for the bubble by setting the visible property of legendSettings as **true** and type property of legendSettings as **Bubbles**.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:bubbleSettings='bubbleSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Bubble, Legend } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        legendSettings: {
            visible: true,
            type: 'Bubbles'
        },
        shapeData: world_map,
        shapeDataPath: 'name',
        shapePropertyPath: 'name',
        bubbleSettings: [{
            visible: true,
            minRadius: 20,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            dataSource: [
                { color: 'green', name: 'India', population: '38332521' },
                { color: 'purple', name: 'New Zealand', population:
'19651127' },
                { color: 'blue', name: 'Pakistan', population: '3090416' }
            ],
            maxRadius: 40,
            colorValuePath: 'color',
            valuePath: 'population'
        }]
    }
},
provide: {
    maps: [Bubble, Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs57" %}

Enable legend for markers

To enable legend for marker by setting the visible property of legendSettings as **true** and type property of legendSettings as **Markers**. The legendText property in the markerSettings can be used to show the legend text based on values provided in the data source.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps :legendSettings='legendSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerSettings='markerSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Legend } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        legendSettings: {
            visible: true,
            type: 'Markers'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        },
        shapeData: world_map,
        markerSettings: [
            {
                visible: true,
                legendText: 'name',
                dataSource: [
                    { latitude: 37.6276571, longitude: -122.4276688, name:
'San Bruno' },
                    { latitude: 33.5302186, longitude: -117.7418381, name:
'Laguna Niguel' },
                    { latitude: 40.7424509, longitude: -74.0081468, name:
'New York' }
                ],
                shape: 'Circle'
            }
        ]
    }
},
provide: {
    maps: [Legend, Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs58" %}

Imitate/Map marker shape to the legend shape

To imitate or map the marker shape with its legend item shape, set the useMarkerShape property to **true** in the legendSettings property.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps id='container' :legendSettings='legendSettings'>
            <e-layers>
                <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings' :markerSettings='markerSettings'></e-layer>
            </e-layers>
        </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Legend } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
import { markerDataSource } from './marker-data.js';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(MapsPlugin);
export default {
data () {
    return {
        legendSettings: {
            visible: true,
            type: 'Markers',
            useMarkerShape:true,
            toggleLegendSettings: {
                enable: true,
                applyShapeSettings: false,
                border: {
                    color: 'green',
                    width: 2
                }
            }
        },
        shapeSettings: {
            fill: '#E5E5E5'
        },
        shapeData: world_map,
        markerSettings: [
            {
                colorValuePath: 'color',
                shapeValuePath:'shape',
                legendText: 'country',
                visible: true
                dataSource: markerDataSource
            }
        ]
    }
},
provide: {
    maps: [Legend, Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs59" %}

Annotations in Vue Maps component

Annotations are used to mark the specific area of interest in the Maps with texts, shapes, or images. Any number of annotations can be added to the Maps component.

Annotation

By using the content property of e-maps-annotation, text content or id of an element or an HTML string can be specified to render a new HTML element in Maps.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps >
                <e-maps-annotations>
                    <e-maps-annotation :content='contentTemplate' :x='x1'
:y='y1' :zIndex='zindex'>
                    </e-maps-annotation>
                </e-maps-annotations>
                <e-layers>
                    <e-layer :shapeData='shapeData' >
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent, Annotations } from '@syncfusion/ej2-vue-
maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
       shapeData: world_map,
       zindex:1,
       x1:'0%',
       y1:'50%',
       contentTemplate: function () {
          return {
          template: Vue.component('MapsComponent', {
            template: '<div id="first"><h1>Maps</h1></div>',
            data() { return {  }; }
          })
        }
      },
    }
},
provide: {
    maps: [ Annotations ]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Annotation customization

Changing the z-index

The stack order of an annotation element can be changed using the zIndex property in the e-maps-annotation.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-maps-annotations>
                  <e-maps-annotation :content='contentTemplate' :x='x1'
:y='y1' :zIndex='zindex'>
                  </e-maps-annotation>
                </e-maps-annotations>
                <e-layers>
                    <e-layer :shapeData='shapeData' >
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent, Annotations } from '@syncfusion/ej2-vue-
maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
       shapeData: world_map,
       zindex:1,
       x1:'0%',
       y1:'50%',
       contentTemplate: function () {
          return {
          template: Vue.component('MapsComponent', {
            template: '<div id="first"><h1>Maps</h1></div>',
            data() { return {  }; }
          })
        }
      },
    }
},
provide: {
    maps: [ Annotations ]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
```

```
   }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs2" %}

Positioning an annotation

Annotations can be placed anywhere in the Maps by specifying pixel or percentage values to the x and y properties in the e-maps-annotation.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-maps-annotations>
                  <e-maps-annotation :content='contentTemplate' :x='x1'
:y='y1' :zIndex='zindex'>
                  </e-maps-annotation>
                </e-maps-annotations>
                <e-layers>
                    <e-layer :shapeData='shapeData' >
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent, Annotations } from '@syncfusion/ej2-vue-
maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
       shapeData: world_map,
       zindex:1,
       x1:'20%',
       y1:'50%',
       contentTemplate: function () {
          return {
          template: Vue.component('MapsComponent', {
            template: '<div id="first"><h1>Maps</h1></div>',
            data() { return {  }; }
          })
        }
      },
    }
},
provide: {
    maps: [ Annotations ]
}
}
</script>
```

```
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs3" %}

Alignment of an annotation

Annotations can be aligned using the horizontalAlignment and verticalAlignment properties in the e-maps-annotation. The possible values can be **Center**, **Far**, **Near** and **None**.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-maps-annotations>
                  <e-maps-annotation :content='contentTemplate' :x='x1'
:y='y1' :zIndex='zindex'
                  horizontalAlignment='Center' verticalAlignment='Center'>
                  </e-maps-annotation>
                </e-maps-annotations>
                <e-layers>
                    <e-layer :shapeData='shapeData' >
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent, Annotations } from '@syncfusion/ej2-vue-
maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
       shapeData: world_map,
       zindex:1,
       x1:'20%',
       y1:'50%',
       contentTemplate: function () {
          return {
          template: Vue.component('MapsComponent', {
            template: '<div id="first"><h1>Maps</h1></div>',
            data() { return {  }; }
          })
        }
      },
    }
},
```

```
provide: {
    maps: [ Annotations ]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs4" %}

Multiple Annotation

Multiple annotations can be added to the Maps by adding multiple e-maps-annotation in the e-maps-annotation and customization for the annotations can be done with the e-maps-annotation.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-maps-annotations>
                  <e-maps-annotation :content='contentTemplate' :x='x1'
:y='y1' :zIndex='zindex'>
                  </e-maps-annotation>
                   <e-maps-annotation :content='contentTemplate1' :x='x2'
:y='y2' :zIndex='zindex1'>
                  </e-maps-annotation>
                </e-maps-annotations>
                <e-layers>
                    <e-layer :shapeData='shapeData' >
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent, Annotations } from '@syncfusion/ej2-vue-
maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
       shapeData: world_map,
       zindex:1,
       x1:'50%',
       y1:'0%',
       zindex1:1,
       x2: '20%',
       y2: '50%',
```

```
        contentTemplate: function () {
            return {
            template: Vue.component('MapsComponent', {
              template: '<div id="first"><h1>Maps</h1></div>',
              data() { return {  }; }
            })
          }
        },
        contentTemplate1: function () {
            return {
            template: Vue.component('MapsComponent', {
              template: '<div id="first"><h1>Multiple-annotation</h1></div>',
              data() { return {  }; }
            })
          }
        }
      }
},
provide: {
    maps: [ Annotations ]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs5" %}

Navigation line in Vue Maps component

The navigation lines are used to denote the path between two locations. This feature can be used to draw flight or sea routes. Navigation lines are enabled by setting the visible property of the navigationLineSettings to **true**.

Customization

The following properties are available in navigationLineSettings to customize the navigation line of the Maps component.

- color - To apply the color for navigation lines in Maps.
- dashArray - To define the pattern of dashes and gaps that is applied to the outline of the navigation lines.
- width - To customize the width of the navigation lines.
- angle - To customize the angle of the navigation lines.
- highlightSettings - To customize the highlight settings of the navigation line.
- selectionSettings - To customize the selection settings of the navigation line.

To navigate the line between two cities on the world map, latitude and longitude values are used to indicate the start and end points of navigation lines drawn on Maps.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData' >
                    <e-navigationLineSettings>
                    <e-navigationLineSetting visible = true :latitude
='latitude' :longitude ='longitude' :color ='color' :angle ='angle'
:width="width" :dashArray='dashArray' >
                    </e-navigationLineSetting>
                    </e-navigationLineSettings>
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, NavigationLine } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: world_map,
        latitude: [40.7128, 36.7783],
        longitude: [-74.0060, -119.4179],
        color: 'black',
        angle: 90,
        width: 2,
        dashArray: '4'
    }
},
provide: {
    maps: [NavigationLine]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs73" %}

Enabling the arrows

To enable the arrow in the navigation line, set the showArrow property of arrowSettings to **true**. The following properties are available in arrowSettings to customize the arrow of the navigation lines.

- color - To apply the color for arrow of the navigation line.
- offset - To customize the offset position of the arrow of the navigation line.

Copyright © 2001 -2024 Syncfusion Inc.

- position - To customize the position of the arrow in navigation line. The possible values can be **Start** and **End**.
- size - To customize the size of the arrow in pixels.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData' >
                    <e-navigationLineSettings>
                    <e-navigationLineSetting visible = true :latitude
='latitude' :longitude ='longitude' :color ='color' :angle ='angle'
:width="width" :dashArray='dashArray' :arrowSettings = 'arrowSettings' >
                    </e-navigationLineSetting>
                    </e-navigationLineSettings>
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, NavigationLine } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        latitude: [37.6276571, -14.2350],
        longitude: [-74.0060, -51.9253],
        color: 'black',
        angle: -180,
        width: 2,
        dashArray: '4',
        arrowSettings: {
            showArrow: true,
            size: 10,
            position: 'Start'
        },
        shapeData: world_map,
    }
},
provide: {
    maps: [NavigationLine]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs74" %}

User interactions in Vue Maps component

Zooming

The zooming feature is used to zoom in and out of Maps to show in-depth information. It is controlled by the zoomFactor property of the zoomSettings. The zoomFactor is increased or decrease dynamically based on zoom in and out interaction.

Enable zooming

Zooming of Maps is enabled by setting the enable property of zoomSettings to **true**.

Enable panning

To enable the panning feature, set the enablePanning property of zoomSettings to **true**.

`

<template>

<ejs-maps :zoomSettings='zoomSettings' >

<e-layers>

<e-layer :shapeData='shapeData' ></e-layer>

</e-layers>

</ejs-maps>

</template>

<script>

import Vue from 'vue';

import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';

import { world_map } from './world-map.js';

Vue.use(MapsPlugin);

export default {

data () {

return{

zoomSettings: {

enable: true,

Copyright © 2001 -2024 Syncfusion Inc.

```
enablePanning: true
},
shapeData: world_map,
}
},
provide: {
maps: [Zoom]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Various type of zooming

Zooming can be performed in following types:

Zooming toolbar

By default, the toolbar is rendered with **zoom-in**, **zoom-out**, and **reset** options when it is set to **true** in the enable property of zoomSettings.

The following options are available in toolbar.

1. Zoom - Provides rectangular zoom support.
2. ZoomIn - Zoom in the Maps.
3. ZoomOut - Zoom out the Maps.
4. Pan - Switches to panning if rectangular zoom is activated.
5. Reset - Restores the Maps to the default view.

```
`
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps :zoomSettings='zoomSettings' >
<e-layers>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-layer :shapeData='shapeData' ></e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
enable: true,
buttonSettings: {
toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset']
}
},
shapeData: world_map,
}
},
provide: {
maps: [Zoom]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
`
```

Pinch zooming

To enable or disable the pinch zooming, use the pinchZooming property in zoomSettings.

```
`
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps :zoomSettings='zoomSettings' >
<e-layers>
<e-layer :shapeData='shapeData' ></e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
enable: true,
pinchZooming: true
},
shapeData: world_map,
}
},
provide: {
maps: [Zoom]
```

```
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Single-click zooming

To enable or disable the single-click zooming, use the zoomOnClick property in zoomSettings.

```
`
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps :zoomSettings='zoomSettings' >
<e-layers>
<e-layer :shapeData='shapeData' ></e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
```

```
enable: true,
zoomOnClick: true
},
shapeData: world_map,
}
},
provide: {
maps: [Zoom]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Double-click zooming

To enable or disable the double-click zooming, use the doubleClickZoom property in zoomSettings.

```
`
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps :zoomSettings='zoomSettings' >
<e-layers>
<e-layer :shapeData='shapeData' ></e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
enable: true,
doubleClickZoom: true
},
shapeData: world_map,
}
},
provide: {
maps: [Zoom]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Mouse wheel zooming

To enable or disable mouse wheel zooming, use the mouseWheelZoom property in zoomSettings.

```
`
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps :zoomSettings='zoomSettings' >
```

```
<e-layers>
<e-layer :shapeData='shapeData' ></e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
enable: true,
mouseWheelZoom: true
},
shapeData: world_map,
}
},
provide: {
maps: [Zoom]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

`

Selection zooming

To enable or disable selection zooming, use the enableSelectionZooming property in zoomSettings. The enablePanning property must be set to **false** to enable the selection zooming in Maps.

`

```
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps :zoomSettings='zoomSettings' >
<e-layers>
<e-layer :shapeData='shapeData' ></e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
enable: true,
enableSelectionZooming: true,
enablePanning: false
},
shapeData: world_map,
}
},
provide: {
maps: [Zoom]
```

```
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Setting minimum and maximum values for zoom factor

The zooming range can be adjusted using the minZoom and maxZoom properties in zoomSettings. The minZoom value is set to 1 by default, and the maxZoom value is set to 10.

```
`
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps :zoomSettings='zoomSettings' >
<e-layers>
<e-layer :shapeData='shapeData' ></e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
```

```
enable: true,
minZoom: 2,
maxZoom: 12
},
shapeData: world_map,
}
},
provide: {
maps: [Zoom]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Zooming with animation

To zoom in or zoom out the Maps with animation, use the animationDuration property in layers.

```
`
<template>
<div id="app">
<div class='wrapper'>
<ejs-maps :zoomSettings='zoomSettings' >
<e-layers>
<e-layer :shapeData='shapeData' :animationDuration='animationDuration' ></e-layer>
</e-layers>
</ejs-maps>
</div>
</div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
return{
zoomSettings: {
enable: true,
mouseWheelZoom: true
},
shapeData: world_map,
animationDuration: 500
}
},
provide: {
maps: [Zoom]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :zoomSettings='zoomSettings' >
                <e-layers>
                    <e-layer :shapeData='shapeData' ></e-layer>
                </e-layers>
```

```
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        zoomSettings: {
            enable: true,
        },
        shapeData: world_map,
    }
},
provide: {
    maps: [Zoom]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs87" %}

Customizing the zoom toolbar

The zoom toolbar can be customized by using the toolbarSettings option in the zoomSettings. The following properties can be used to customize the zoom toolbar.

- backgroundColor - It is used to customize the background color of the zoom toolbar.
- borderOpacity - It is used to customize the opacity of the border of the zoom toolbar.
- borderWidth - It is used to customize the thickness of the border of the zoom toolbar.
- borderColor - It is used to customize the color of the border of the zoom toolbar.
- horizontalAlignment - It is used to position the zoom toolbar in near, far, and center positions to customize its horizontal placement.
- verticalAlignment - It is used to position the zoom toolbar in near, far, and center positions to customize its vertical placement.
- orientation - It is used to change the orientation (horizontal/vertical) of the zoom toolbar.

APP.VUE

```
<template>
    <div id="app">
          <div>
            <ejs-maps :zoomSettings='zoomSettings'>
                <e-layers>
```

```
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom} from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeSettings:{
          fill:'#C1DFF5'
        },
        zoomSettings: {
        enable: true,
            toolbarSettings:{
                orientation:'Vertical',
                backgroundColor:'pink',
                borderWidth:3,
                borderColor:'green',
                verticalAlignment:'Near',
                buttonSettings: {
                   toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan',
'Reset']
                }
            }
        }
    }
},
provide: {
    maps: [Zoom]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs102" %}

<!-- markdownlint-disable MD036 -->

Customizing the buttons in the zoom toolbar

The appearance of the buttons in the zoom toolbar can be customized by using the buttonSettings option in the toolbarSettings of the zoomSettings property. The following properties can be used to customize the zoom toolbar buttons.

- fill - It is used to set the background color of the buttons.
- color - It is used to customize the color of the icons inside the button.
- borderOpacity - It is used to set the opacity of the border of the zoom toolbar buttons.
- borderWidth - It is used to set the thickness of the border of the zoom toolbar buttons.

Copyright © 2001 -2024 Syncfusion Inc.

- borderColor - It is used to set the color of the border of the zoom toolbar buttons.
- radius - It is used to set the size of the button.
- selectionColor - It is used to set the color of the icons inside the button when selection is performed.
- highlightColor - It is used to change the color of the button when the mouse is hovered over it.
- padding - It is used to change the padding space between each button.
- opacity - It is used to change the opacity of the button.
- toolbarItems - It is used to change the items that should be displayed in the zoom toolbar. By default, zoom-in, zoom-out, and reset buttons will be available. Other options include selection zoom and panning.

APP.VUE

```
<template>
    <div id="app">
          <div>
            <ejs-maps :zoomSettings='zoomSettings'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom} from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeSettings:{
          fill:'#C1DFF5'
        },
        zoomSettings: {
        enable: true,
            toolbarSettings:{
                buttonSettings: {
                   fill:'pink',
                   padding: 10,
                   toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan',
'Reset'],
                   color: 'red',
                   borderColor:'green',
                   radius:35,
                   selectionColor:'#d55e5e',
                   hightlightColor:'#5ed59a',
                   opacity:0.6,
                   borderWidth: 2
                }
            }
        }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
},
provide: {
    maps: [Zoom]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs103" %}

<!-- markdownlint-disable MD036 -->

Customizing the tooltip of the zoom toolbar

The appearance of the tooltip of the zoom toolbar can be customized by using the tooltipSettings option in the toolbarSettings of the zoomSettings property. The following properties are available to customize the zoom toolbar tooltip.

- visible - Enables or disables the tooltip of the zoom toolbar.
- fill - It is used to change the background color of the tooltip of the zoom toolbar.
- borderOpacity - It is used to change the opacity of the border of the zoom toolbar's tooltip.
- borderWidth - It is used to change the thickness of the border of the zoom toolbar's tooltip.
- borderColor - It is used to change the color of the border of the zoom toolbar's tooltip.
- fontColor - It is used to change the color of the text in the tooltip of the zoom toolbar.
- fontFamily - It is used to change the font family of the text in the tooltip of the zoom toolbar.
- fontStyle - It is used to change the font style of the text in the tooltip of the zoom toolbar.
- fontWeight - It is used to change the font weight of the text in the tooltip of the zoom toolbar.
- fontSize - It is used to change the size of the text in the tooltip of the zoom toolbar.
- fontOpacity - It is used to change the opacity of the text in the tooltip of the zoom toolbar.

APP.VUE

```
<template>
    <div id="app">
          <div>
            <ejs-maps :zoomSettings='zoomSettings'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom} from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapeSettings:{
```

```
            fill:'#C1DFF5'
        },
        zoomSettings: {
        enable: true,
            toolbarSettings:{
                tooltipSettings:{
                    visible:true,
                    borderWidth:2,
                    borderColor:'green',
                    fontColor:'black',
                    fill:'violet',
                    fontFamily:'Times New Roman',
                    fontWeight:200,
                    fontSize:'22px',
                    fontOpacity:1
                },
                buttonSettings: {
                   toolbarItems: ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan',
'Reset']
                }
            }
        }
    }
},
provide: {
    maps: [Zoom]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs104" %}

Selection

Each shape in the Maps can be selected and deselected during interaction with the shapes. Selection is enabled by setting the enable property of selectionSettings to **true**.

The following properties are available to customize the selection of Maps elements such as shapes, bubbles, markers and legends.

- border - To customize the color, width and opacity of the border of which element is selected in Maps.
- fill - Applies the color for the element that is selected.
- opacity - To customize the transparency for the element that is selected.
- enableMultiSelect - To enable or disable the selection for multiple shapes or markers or bubbles in the Maps.

By tapping on the specific legend, the shapes which are bounded to the selected legend is also selected and vice versa.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <ejs-maps :layers='layers' :legendSettings='legendSettings'>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Selection, Legend } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
         layers: [{
            shapeData: world_map,
            dataSource: [{  "Country": "China", "Membership": "Permanent"},
            {"Country": "France","Membership": "Permanent" },
            { "Country": "Russia","Membership": "Permanent"},
            {"Country": "Kazakhstan","Membership": "Non-Permanent"},
            { "Country": "Poland","Membership": "Non-Permanent"},
            {"Country": "Sweden","Membership": "Non-Permanent"}],
            shapePropertyPath: 'name',
            shapeDataPath: 'Country',
            selectionSettings: {
               enable: true,
               fill: 'blue',
               border: { color: 'white', width: 2}
            },
            shapeSettings: {
                colorValuePath: 'Membership',
                colorMapping: [
                {
                    value: 'Permanent', color: '#D84444'
                },
                {
                    value: 'Non-Permanent', color: '#316DB5'
                }]
            }
        }],
    legendSettings: {
        visible: true
    }
    }
},
provide: {
    maps: [Selection, Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs88" %}

Enable selection for bubbles

To enable the selection for bubbles in Maps, set the selectionSettings in bubbleSettings and set the enable property of selectionSettings as **true**.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :layers='layers'>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Selection, Bubble } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        layers: [{
            shapeData: world_map,
            shapeDataPath: 'name',
            shapePropertyPath: 'name',
            bubbleSettings: [{
                visible: true,
                dataSource: [
                    { name: 'India', population: '38332521' },
                    { name: 'South Africa', population: '19651127' },
                    { name: 'Pakistan', population: '3090416' }
                ],
            selectionSettings: {
                enable: true,
                fill: 'green',
                border: { color: 'white', width: 2}
            },
            valuePath: 'population'
        }]
    }]
    }
},
provide: {
    maps: [Selection, Bubble]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs89" %}

Enable selection for markers

To enable the selection for markers in Maps, set the selectionSettings in themarkerSettings and set the enable property of the selectionSettings as **true**.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :layers='layers'>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Selection, Marker } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        layers: [{
            shapeData: world_map,
            markerSettings: [{
                visible: true,
                height: 20,
                width: 20,
                fill: 'green',
                shape:'Balloon',
                selectionSettings: {
                    enable: true,
                    fill: 'blue',
                    border: { color: 'white', width: 2}
                },
                dataSource: [
                    { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'Europe' },
                    { latitude: 59.88893689676585, longitude: -109.3359375,
name:'North America'},
                    { latitude: -6.64607562172573, longitude: -
55.54687499999999, name:'South America'}
                ]
            }],
        }]
    }
},
provide: {
    maps: [Selection, Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs90" %}

Enable selection for polygons

When the enable property of selectionSettings is set to **true**, the polygon shapes can be selected via user interaction. The following properties are available in selectionSettings to customize the polygon shape when it is selected.

- enableMultiSelect - It is used to enable multiple selection of polygon shapes.
- fill - It is used to change the color of the selected polygon shape.
- opacity - It is used to change the opacity of the selected polygon shape.
- border - This property is used to change the color, width, and opacity of the border of the selected polygon shape.

To use the polygon feature, the **Polygon** module must be injected, as described in this link.

The following example shows how to select the polygon shape in the geometry map.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:polygonSettings='polygonSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Polygon, Highlight, Selection } from '@syncfusion/ej2-
vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
    data() {
        return {
            shapeData: world_map,
            polygonSettings: {
                polygons: [{
                        points: [
                            { longitude: -1.8920678947185365, latitude:
35.06195799239681 },
                            { longitude: -1.6479633699113947, latitude:
33.58989612266137 },
                            { longitude: -1.4201220366858252, latitude:
32.819439646045254 },
```

```
                        { longitude: -1.197974596225663, latitude:
32.26940895444655 },
                        { longitude: -2.891112397949655, latitude:
32.10303058820031 },
                        { longitude: -3.8246984550501963, latitude:
31.34551662687602 },
                        { longitude: -3.720166273688733, latitude:
30.758086682848685 },
                        { longitude: -5.6571886081189575, latitude:
29.613582597203006 },
                        { longitude: -7.423353242214745, latitude:
29.44328441403087 },
                        { longitude: -8.6048931685323, latitude:
28.761444633616776 },
                        { longitude: -8.695726975465703, latitude:
27.353491085576195 },
                        { longitude: 3.837867279970908, latitude:
19.15916564839422 },
                        { longitude: 6.0705408799045415, latitude:
19.48749097192868 },
                        { longitude: 12.055736352807713, latitude:
23.694596786078293 },
                        { longitude: 11.272522332402986, latitude:
24.289329186946034 },
                        { longitude: 10.30872578261932, latitude:
24.65419958524693 },
                        { longitude: 9.910236690050027, latitude:
25.48943950947175 },
                        { longitude: 9.432639882414293, latitude:
26.398372489836902 },
                        { longitude: 9.898266456582292, latitude:
26.73489453809293 },
                        { longitude: 9.560243026853641, latitude:
30.31040379467153 },
                        { longitude: 8.943853847283322, latitude:
32.350324876652195 },
                        { longitude: 7.57004059025715, latitude:
33.75071049019398 },
                        { longitude: 8.0906322609153, latitude:
34.69043151009983 },
                        { longitude: 8.363285449347273, latitude:
35.38654406371319 },
                        { longitude: 8.26139549449448, latitude:
36.44751078733985 },
                        { longitude: 8.61100824823302, latitude:
36.881913362940196 },
                        { longitude: 7.4216488925819135, latitude:
37.021408008916254 },
                        { longitude: 6.461182254165351, latitude:
36.99092409199429 },
                        { longitude: 5.297178918070159, latitude:
36.69985479014656 },
                        { longitude: 3.6718056161224695, latitude:
36.86470546831693 },
                        { longitude: 1.2050052555659931, latitude:
36.57658056301722 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                                { longitude: -0.26968570003779746, latitude:
35.806903541813625 },
                                { longitude: -0.995191786435754, latitude:
35.58466127904214 },
                                { longitude: -1.8920678947185365, latitude:
35.06195799239681 },
                            ],
                            fill: 'blue',
                            opacity: 0.7,
                            borderColor: 'green',
                            borderWidth: 2,
                            borderOpacity: 0.7,
                        }],
                    highlightSettings: {
                        enable: true,
                        fill: 'blue',
                        opacity: 0.7,
                        border: {
                            color: 'green',
                            width: 2,
                            opacity: 0.7
                        }
                    },
                    selectionSettings: {
                        enable: true,
                        fill: 'violet',
                        enableMultiSelect: false,
                        opacity: 0.8,
                        border: {
                            color: 'cyan',
                            opacity: 1,
                            width: 7
                        }
                    }
                }
            }
        },
        provide: {
            maps: [Polygon, Highlight, Selection]
        },
}
</script>
<style>
.wrapper {
    max-width: 400px;
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/polygon-cs2" %}

Public method for the shape selection

The shapeSelection method can be used to select each shape in the Maps.

LayerIndex, propertyName, country name, and selected value as a boolean state(true / false) are the input parameters for this method.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps ref='maps'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:selectionSettings='selectionSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
        <ejs-button id='selection' @click.native='select'>selection</ejs-
button>
        <ejs-button id='unselection'
@click.native='unselect'>unselection</ejs-button>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Selection } from '@syncfusion/ej2-vue-maps';
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
Vue.use(ButtonPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        selectionSettings: {
            enable: true,
            fill: 'green',
            border: { color: 'white', width: 2}
        }
    }
},
provide: {
    maps: [Selection]
},
methods:{
    select:function(){
        this.$refs.maps.shapeSelection(0, "continent", "Asia", true);
    },
    unselect:function(){
        this.$refs.maps.shapeSelection(0, "continent", "Asia", false);
    }
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs91" %}

Initial shape selection

The shape is initially selected using the initialShapeSelection, and the values are mapped to the shapePath and shapeValue.

Note: initialShapeSelection is an Array property.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps  >
                <e-layers>
                    <e-layer :shapeData='shapeData' :selectionSettings=
'selectionSettings' :initialShapeSelection= 'initialShapeSelection'></e-
layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Legend, Selection } from '@syncfusion/ej2-vue-
maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        selectionSettings: {
            enable: true,
            fill: 'green',
            border: { color: 'white', width: 2}
        },
        shapeData: world_map,
        initialShapeSelection: [
            { shapePath: 'continent', shapeValue: 'Africa' },
            { shapePath: 'name', shapeValue: 'India' }
        ],
    }
},
provide: {
    maps: [Selection, Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs92" %}

Initial marker selection

Using the initialMarkerSelection, the marker shape can be selected initially. Markers render based on the latitude and longitude values.

Note: initialMarkerSelection is an Array property.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps  >
                <e-layers>
                    <e-layer :shapeData='shapeData' :markerSettings=
'markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Selection } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: world_map,
        markerSettings: [{
            visible: true,
            height: 20,
            width: 20,
            fill: 'green',
            shape:'Balloon',
            initialMarkerSelection: [{
               latitude: -6.64607562172573, longitude: -55.54687499999999
            }],
            selectionSettings: {
                enable: true,
                fill: 'blue',
                opacity: 1,
                border: { color: 'white', width: 2, opacity: 1}
            },
            dataSource: [
                { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'Europe' },
                { latitude: 59.88893689676585, longitude: -109.3359375,
name:'North America'},
                { latitude: -6.64607562172573, longitude: -
55.54687499999999, name:'South America'}
            ]
        }]
    }
},
```

```
provide: {
    maps: [Selection, Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs93" %}

Highlight

Each shape in the Maps can be highlighted during mouse hover on the Maps elements such as shapes, bubbles, markers and legends. Highlight is enabled by setting the enable property of highlightSettings to **true**.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :layers='layers' :legendSettings='legendSettings'>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Highlight, Legend } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        layers: [
        {
            shapeData: world_map,
            dataSource: [
                { "Country": "China", "Membership": "Permanent"},
                { "Country": "France", "Membership": "Permanent" },
                { "Country": "Russia", "Membership": "Permanent"},
                { "Country": "Kazakhstan", "Membership": "Non-Permanent"},
                { "Country": "Poland", "Membership": "Non-Permanent"},
                { "Country": "Sweden", "Membership": "Non-Permanent"}
            ],
            shapePropertyPath: 'name',
            shapeDataPath: 'Country',
            highlightSettings: {
               enable: true,
               fill: 'blue',
               border: { color: 'white', width: 2}
            },
            shapeSettings: {
```

```
                colorValuePath: 'Membership',
                colorMapping: [
                {
                    value: 'Permanent', color: '#D84444'
                },
                {
                    value: 'Non-Permanent', color: '#316DB5'
                }]
            }
        }
    ],
    legendSettings: {
        visible: true
    }
    }
},
provide: {
    maps: [Highlight, Legend]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs94" %}

Enable highlight for bubbles

To enable the highlight for bubbles in Maps, set the highlightSettings in bubbleSettings and set the enable property of highlightSettings as **true**.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :layers='layers'>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Highlight, Bubble } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        layers: [{
            shapeData: world_map,
            shapeDataPath: 'name',
            shapePropertyPath: 'name',
```

```
            bubbleSettings: [{
                visible: true,
                dataSource: [
                    { name: 'India', population: '38332521' },
                    { name: 'South Africa', population: '19651127' },
                    { name: 'Pakistan', population: '3090416' }
                ],
            highlightSettings: {
                enable: true,
                fill: 'green',
                border: { color: 'white', width: 2}
            },
            valuePath: 'population'
        }]
    }]
    }
},
provide: {
    maps: [Highlight, Bubble]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs95" %}

Enable highlight for markers

To enable the highlight for markers in Maps, set the highlightSettings in markerSettings and set the enable property of highlightSettings as **true**.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :layers='layers'>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Highlight, Marker } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        layers: [{
            shapeData: world_map,
            markerSettings: [{
```

```
                visible: true,
                height: 20,
                width: 20,
                fill: 'green',
                shape:'Balloon',
                highlightSettings: {
                    enable: true,
                    fill: 'blue',
                    border: { color: 'white', width: 2}
                },
            dataSource: [
                { latitude: 49.95121990866204, longitude:
18.468749999999998, name:'Europe' },
                { latitude: 59.88893689676585, longitude: -109.3359375,
name:'North America'},
                { latitude: -6.64607562172573, longitude: -
55.54687499999999, name:'South America'}
            ]
        }],
    }]
    }
},
provide: {
    maps: [Highlight, Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs96" %}

Enable highlight for polygons

The polygon shapes can be highlighted via user interaction if the enable property of highlightSettings is set to **true**. The following properties are available in highlightSettings to customize the polygon shape when it is highlighted.

- fill - It is used to change the color of the highlighted polygon shape.
- opacity - It is used to change the opacity of the highlighted polygon shape.
- border - This property is used to change the color, width, and opacity of the border of the highlighted polygon shape.

To use the polygon feature, the **Polygon** module must be injected, as described in this link.

The following example shows how to highlight a polygon shape on a geometry map.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
```

```
            <ejs-maps>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:polygonSettings='polygonSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Polygon, Highlight, Selection } from '@syncfusion/ej2-
vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
    data() {
        return {
            shapeData: world_map,
            polygonSettings: {
                polygons: [{
                        points: [
                            { longitude: -1.8920678947185365, latitude:
35.06195799239681 },
                            { longitude: -1.6479633699113947, latitude:
33.58989612266137 },
                            { longitude: -1.4201220366858252, latitude:
32.819439646045254 },
                            { longitude: -1.197974596225663, latitude:
32.26940895444655 },
                            { longitude: -2.891112397949655, latitude:
32.10303058820031 },
                            { longitude: -3.8246984550501963, latitude:
31.34551662687602 },
                            { longitude: -3.720166273688733, latitude:
30.758086682848685 },
                            { longitude: -5.6571886081189575, latitude:
29.613582597203006 },
                            { longitude: -7.423353242214745, latitude:
29.44328441403087 },
                            { longitude: -8.6048931685323, latitude:
28.761444633616776 },
                            { longitude: -8.695726975465703, latitude:
27.353491085576195 },
                            { longitude: 3.837867279970908, latitude:
19.15916564839422 },
                            { longitude: 6.0705408799045415, latitude:
19.48749097192868 },
                            { longitude: 12.055736352807713, latitude:
23.694596786078293 },
                            { longitude: 11.272522332402986, latitude:
24.289329186946034 },
                            { longitude: 10.30872578261932, latitude:
24.65419958524693 },
                            { longitude: 9.910236690050027, latitude:
25.48943950947175 },
```

```
                                { longitude: 9.432639882414293, latitude:
26.398372489836902 },
                                { longitude: 9.898266456582292, latitude:
26.73489453809293 },
                                { longitude: 9.560243026853641, latitude:
30.31040379467153 },
                                { longitude: 8.943853847283322, latitude:
32.350324876652195 },
                                { longitude: 7.57004059025715, latitude:
33.75071049019398 },
                                { longitude: 8.0906322609153, latitude:
34.69043151009983 },
                                { longitude: 8.363285449347273, latitude:
35.38654406371319 },
                                { longitude: 8.26139549449448, latitude:
36.44751078733985 },
                                { longitude: 8.61100824823302, latitude:
36.881913362940196 },
                                { longitude: 7.4216488925819135, latitude:
37.021408008916254 },
                                { longitude: 6.461182254165351, latitude:
36.99092409199429 },
                                { longitude: 5.297178918070159, latitude:
36.69985479014656 },
                                { longitude: 3.6718056161224695, latitude:
36.86470546831693 },
                                { longitude: 1.2050052555659931, latitude:
36.57658056301722 },
                                { longitude: -0.26968570003779746, latitude:
35.806903541813625 },
                                { longitude: -0.995191786435754, latitude:
35.58466127904214 },
                                { longitude: -1.8920678947185365, latitude:
35.06195799239681 },
                            ],
                            fill: 'red',
                            opacity: 0.7,
                            borderColor: 'green',
                            borderWidth: 2,
                            borderOpacity: 0.7,
                        }],
                    highlightSettings: {
                        enable: true,
                        fill: 'yellow',
                        opacity: 0.4,
                        border: {
                            color: 'blue',
                            opacity: 0.6,
                            width: 4
                        }
                    },
                    selectionSettings: {
                        enable: true,
                        fill: 'red',
                        opacity: 0.7,
                        border: {
                            color: 'green',
```

```
                        width: 2,
                        opacity: 0.7
                    }
                }
            }
        }
    },
    provide: {
        maps: [Polygon, Highlight, Selection]
    },
}
</script>
<style>
.wrapper {
    max-width: 400px;
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/polygon-cs3" %}

Tooltip

On mouse over or touch end event, the tooltip is used to get more information about the layer, bubble, or marker. It can be enabled separately for layer or bubble or marker by using the visible property of tooltipSettings as **true**. The valuePath property in the tooltip takes the field name that presents in dataSource and displays that value as tooltip text. The tooltipDisplayMode property is used to change the display mode of the tooltip in Maps. Following display modes of tooltip are available in the Maps component. By default, tooltipDisplayMode is set to **MouseMove**.

- MouseMove
- Click
- DoubleClick

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:tooltipSettings='tooltipSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsTooltip } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    return{
        shapeData: world_map,
        tooltipSettings: {
            visible: true,
            valuePath: 'name'
        }
    }
},
provide: {
    maps: [MapsTooltip]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs97" %}

Customization

The following properties are available in the tooltipSettings to customize the tooltip of the Maps component.

- border - To customize the color, width and opacity of the border of the tooltip in layers, markers, and bubbles of Maps.
- fill - Applies the color of the tooltip in layers, markers, and bubbles of Maps.
- format - To customize the format of the tooltip in layers, markers, and bubbles of Maps
- textStyle - To customize the style of the text in the tooltip for layers, markers, and bubbles of Maps.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :tooltipRender='tooltipRender'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:shapeSettings='shapeSettings' :dataSource='dataSource'
:tooltipSettings='tooltipSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsTooltip } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
data () {
    return{
        shapeData: world_map,
        shapePropertyPath: 'name',
        shapeDataPath: 'name',
        shapeSettings: {
            fill: '#E5E5E5',
            colorMapping: [
                { color: '#b3daff', value: '1' },
                { color: '#80c1ff', value: '2' },
                { color: '#1a90ff', value: '3' },
                { color: '#005cb3', value: '7' }
            ],
            colorValuePath: 'value1'
        },
        dataSource: [
            { "name": "India", "value1": "3", "value2": "2", "country":
"India" },
            { "name": "Dominican Rep.", "value1": "3", "value2": "2",
"country": "West Indies" },
            { "name": "Cuba", "value1": "3", "value2": "2", "country": "West
Indies" },
            { "name": "Jamaica", "value1": "3", "value2": "2", "country":
"West Indies" },
            { "name": "Haiti", "value1": "3", "value2": "2", "country":
"West Indies" },
            { "name": "Guyana", "value1": "3", "value2": "2", "country":
"West Indies" },
            { "name": "Suriname", "value1": "3", "value2": "2", "country":
"West Indies"},
            { "name": "Trinidad and Tobago", "value1": "3", "value2": "2",
"country": "West Indies" },
            { "name": "Sri Lanka", "value1": "3", "value2": "1", "country":
"Sri Lanka" },
            { "name": "United Kingdom", "value1": "3", "value2": "0",
"country": "England" },
            { "name": "Pakistan", "value1": "2", "value2": "1", "country":
"Pakistan" },
            { "name": "New Zealand", "value1": "1", "value2": "0",
"country": "New Zealand" },
            { "name": "Australia", "value1": "7", "value2": "5", "country":
"Australia" }
        ],
        tooltipSettings: {
            visible: true,
            valuePath: 'name',
            format: '${name}: ${value1}',
            fill: '#D0D0D0',
            textStyle: {
                color: 'green',
                fontFamily: 'Times New Roman',
                fontStyle: 'Sans-serif'
            }
        }
    }
},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
provide: {
    maps: [MapsTooltip]
},
methods:{
    tooltipRender:function(args){
        if (!args.options.data) {
            args.cancel = true;
        }
    }
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs98" %}

Tooltip template

The HTML element can be rendered in the tooltip of the Maps using the template property of the tooltipSettings. In the following example, ${value1} and ${value2} are the place holders in the HTML element to display the value1 and value2 values of the corresponding shape.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :tooltipSettings='tooltipSettings' ></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsTooltip } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
import { default_data } from './default-data.js';
Vue.use(MapsPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        shapePropertyPath: 'continent',
        shapeDataPath: 'continent',
        dataSource: default_data,
        tooltipSettings: {
            visible: true,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            valuePath: 'continent',
            template: '<div style="width:60px; text-align:center;
background-color: white; border: 2px solid black; padding-bottom:
10px;padding-top: 10px;padding-left: 10px;padding-right:
10px;"><span>${continent}</span></div>',
            textStyle: {
                color: 'black'
            }
        }
    }
},
provide: {
    maps: [MapsTooltip]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs99" %}

Print in Vue Maps component

Print

The rendered Maps can be printed directly from the browser by calling the print method. To use the print functionality, **Print** module must be injected into the Maps using **provide** section and set the allowPrint property to **true**. The following code snippet demonstrates injecting the Print module into the Maps component.

`

<template>

<ejs-maps id='maps'></ejs-maps>

</template>

<script>

export default ({

data() {

return{ };

},

provide: {

maps: [Print]

}

</script>

});

`

APP.VUE

```
<template>
    <div id="app">
        <button id="print" @click="clickPrint">Print</button>
        <ejs-maps id='container' :allowPrint='allowPrint' ref="maps"
height='450px' width='500px'>
            <e-layers>
                <e-layer :shapeData='shapeData'
:dataLabelSettings='dataLabelSettings' :shapeSettings='shapeSettings'
:tooltipSettings='tooltipSettings' ></e-layer>
            </e-layers>
        </ejs-maps>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsTooltip, DataLabel, Print } from '@syncfusion/ej2-
vue-maps';
import { usMap } from './usa.js';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(MapsPlugin,ButtonPlugin);
export default {
data () {
    return {
        dataLabelSettings: {
            visible: true,
            labelPath: 'name',
            smartLabelMode: 'Trim'
        },
        shapeData: usMap,
        shapeSettings: {
            autofill: true
        },
        tooltipSettings: {
            visible: true,
            valuePath: 'name'
        },
        allowPrint: true
    }
},
provide: {
    maps: [MapsTooltip, DataLabel, Print]
},
methods: {
    clickPrint: function(args) {
        let map=document.getElementById('container');
        map.ej2_instances[0].print();
    }
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs77" %}

Copyright © 2001 -2024 Syncfusion Inc.

Export

Image Export

To use the image export functionality in Maps, **ImageExport** module must be injected into the Maps using **provide** section and set the allowImageExport property to **true**.

`

<template>

<ejs-maps id='maps'></ejs-maps>

</template>

<script>

export default ({

data() {

return{ };

},

provide: {

maps: [ImageExport]

}

</script>

});

`

The rendered Maps can be exported as an image using the export method. The method requires two parameters: image type and file name. The Maps can be exported as an image in the following formats.

- JPEG
- PNG
- SVG

APP.VUE

```
<template>
    <div id="app">
        <button id="export" @click="clickExport">Export</button>
        <ejs-maps  id='container' :allowImageExport='allowImageExport'
ref='maps' height='450px' width='500px'>
            <e-layers>
                <e-layer :shapeData='shapeData'
:markerSettings='markerSettings' :shapeSettings='shapeSettings'></e-layer>
            </e-layers>
        </ejs-maps>
    </div>
</template>
<script>
import Vue from 'vue';
```

```
import { MapsPlugin, MapsTooltip, Marker, ImageExport} from
'@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use( MapsPlugin );
export default {
data () {
    return {
        shapeData: world_map,
        shapeSettings: { fill: 'lightgrey', border: { color: 'black', width:
0.1 } },
        allowImageExport: true,
        markerSettings: [
            {
                animationDuration: 0,
                visible: true,
                dataSource: [
                    { longitude: 116.5703749, latitude: 40.4319077, name:
'The Great Wall of China, China ' },
                    { longitude: 35.4443622, latitude: 30.3284544, name:
'Petra, Jorden' },
                    { longitude: 78.0421552, latitude: 27.1750151, name:
'Taj Mahal, Agra, India' },
                    { longitude: 12.4922309, latitude: 41.8902102, name:
'The Roman Colosseum, Rome, Italy' },
                    { longitude: -88.5677826, latitude: 20.6842849, name:
'The Chichen Itza, Mexico' },
                    { longitude: -72.5449629, latitude: -13.1631412, name:
'Machu Picchu, Peru' },
                    { longitude: -43.2104872, latitude: -22.951916, name:
'Christ Redeemer, Rio de janeiro, Brazil'}
                ],
                shape: 'Balloon',
                fill: '#E13E40',
                height: 20,
                width: 15,
                tooltipSettings: {
                    visible: true,
                    valuePath: 'name'
                }
            }
        ]
    }
},
provide: {
    maps: [Marker, MapsTooltip, ImageExport]
},
methods: {
     clickExport: function(args) {
        let map = document.getElementById('container');
        map.ej2_instances[0].export("PNG", "Maps");
    }
}
};
</script>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs78" %}

Exporting Maps as base 64 string of the file

The image can be exported as base64 string for the JPEG and PNG formats. The rendered Maps can be exported to image as a base64 string using the export method. There are four parameters required: image type, file name, orientation of the exported PDF document which must be set as **null** for image export and finally **allowDownload** which should be set as **false** to return base64 string.

APP.VUE

```
<template>
    <div id="app">
        <button id="export" @click="clickExport">Export</button>
        <ejs-maps  id='container' :allowImageExport='allowImageExport'
ref='maps' height='450px' width='500px'>
            <e-layers>
                <e-layer :shapeData='shapeData'
:markerSettings='markerSettings' :shapeSettings='shapeSettings'></e-layer>
            </e-layers>
        </ejs-maps>
        <div id="data"></data>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsTooltip, Marker, ImageExport} from
'@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use( MapsPlugin );
export default {
data () {
    return {
        shapeData: world_map,
        shapeSettings: { fill: 'lightgrey', border: { color: 'black', width:
0.1 } },
        allowImageExport: true,
        markerSettings: [
            {
                animationDuration: 0,
                visible: true,
                dataSource: [
                    { longitude: 116.5703749, latitude: 40.4319077, name:
'The Great Wall of China, China ' },
                    { longitude: 35.4443622, latitude: 30.3284544, name:
'Petra, Jorden' },
                    { longitude: 78.0421552, latitude: 27.1750151, name:
'Taj Mahal, Agra, India' },
                    { longitude: 12.4922309, latitude: 41.8902102, name:
'The Roman Colosseum, Rome, Italy' },
                    { longitude: -88.5677826, latitude: 20.6842849, name:
'The Chichen Itza, Mexico' },
                    { longitude: -72.5449629, latitude: -13.1631412, name:
'Machu Picchu, Peru' },
                    { longitude: -43.2104872, latitude: -22.951916, name:
'Christ Redeemer, Rio de janeiro, Brazil'}
                ],
                shape: 'Balloon',
                fill: '#E13E40',
```

```
                height: 20,
                width: 15,
                tooltipSettings: {
                    visible: true,
                    valuePath: 'name'
                }
            }
        ]
    }
},
provide: {
    maps: [Marker, MapsTooltip, ImageExport]
},
methods: {
     clickExport: function(args) {
        let map=document.getElementById('container');
        map.ej2_instances[0].export("PNG","Maps",null,false).then((data) =>
{
            document.getElementById('data').innerHTML = data;
        });
    }
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs79" %}

PDF Export

To use the PDF export functionality in Maps, **PdfExport** module must be injected into the Maps using **provide** section and set the allowPdfExport property to **true**.

`

<template>

<ejs-maps id='maps'></ejs-maps>

</template>

<script>

export default ({

data() {

return{ };

},

provide: {

maps: [PdfExport]

}

</script>

});

`

The rendered Maps can be exported as PDF using the export method. The export method requires three parameters: file type, file name and orientation of the PDF document. The orientation setting is optional and **0** indicates portrait and **1** indicates landscape.

APP.VUE

```
<template>
    <div id="app">
        <button id="export" @click="clickExport">export</button>
        <ejs-maps  id='container' :allowPdfExport='allowPdfExport'
ref='maps' height='450px' width='500px'>
            <e-layers>
                <e-layer :shapeData='shapeData'
:markerSettings='markerSettings' :shapeSettings='shapeSettings'></e-layer>
            </e-layers>
        </ejs-maps>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsTooltip, Marker, PdfExport} from '@syncfusion/ej2-
vue-maps';
import { world_map } from './world-map.js';
Vue.use( MapsPlugin );
export default {
data () {
    return {
        shapeData: world_map,
        shapeSettings: { fill: 'lightgrey', border: { color: 'black', width:
0.1 } },
        allowPdfExport: true,
        markerSettings: [
            {
                animationDuration: 0,
                visible: true,
                dataSource: [
                    { longitude: 116.5703749, latitude: 40.4319077, name:
'The Great Wall of China, China ' },
                    { longitude: 35.4443622, latitude: 30.3284544, name:
'Petra, Jorden' },
                    { longitude: 78.0421552, latitude: 27.1750151, name:
'Taj Mahal, Agra, India' },
                    { longitude: 12.4922309, latitude: 41.8902102, name:
'The Roman Colosseum, Rome, Italy' },
                    { longitude: -88.5677826, latitude: 20.6842849, name:
'The Chichen Itza, Mexico' },
                    { longitude: -72.5449629, latitude: -13.1631412, name:
'Machu Picchu, Peru' },
                    { longitude: -43.2104872, latitude: -22.951916, name:
'Christ Redeemer, Rio de janeiro, Brazil'}
                ],
                shape: 'Balloon',
                fill: '#E13E40',
                height: 20,
                width: 15,
                tooltipSettings: {
                    visible: true,
```

```
                    valuePath: 'name'
                }
            }
        ]
    }
},
provide: {
    maps: [Marker, MapsTooltip, PdfExport]
},
methods: {
     clickExport: function(args) {
        let map=document.getElementById('container');
        map.ej2_instances[0].export("PDF", "Maps", 0);
    }
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs80" %}

The exporting of the Maps as base64 string is not supported for the PDF export.

Export the tile Maps

The rendered Maps with providers such as OSM, Bing and Google static Maps can be exported using the export method. It supports the following export formats.

- JPEG
- PNG
- PDF

APP.VUE

```
<template>
    <div id="app">
        <button id="print" @click="clickPrint">Print</button>
        <button id="export" @click="clickExport">export</button>
        <ejs-maps  id='container' :allowPdfExport='allowPdfExport'
        :allowPrint='allowPrint' :allowImageExport='allowImageExport'
ref='maps' :titleSettings='titleSettings'>
            <e-layers>
                <e-layer :urlTemplate='urlTemplate'></e-layer>
            </e-layers>
        </ejs-maps>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, ImageExport, PdfExport, Print } from '@syncfusion/ej2-
vue-maps';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
Vue.use( MapsPlugin, ButtonPlugin);
enableRipple(true);
export default {
data () {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    return {
        urlTemplate: 'https://tile.openstreetmap.org/level/tileX/tileY.png',
        titleSettings: {
            text: 'OSM'
        },
        allowImageExport: true,
        allowPdfExport: true,
        allowPrint: true
    }
},
provide: {
    maps: [ImageExport, PdfExport, Print]
},
methods: {
    clickExport: function(args) {
      this.$refs.maps.ej2Instances.export("PNG","GAUGE");
    },
    clickPrint:function(args){
        this.$refs.maps.ej2Instances.print();
    }
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs81" %}

State persistence in Vue Maps component

State Persistence

State persistence allows the Maps to retain the current model value in the browser cookies for state maintenance. This action is handled through the enablePersistence property which is set to **false** by default. When this property is set to **true**, some of the Maps component model values are preserved even after the page is refreshed.

`

<template>

<ejs-maps :zoomSettings='zoomSettings' enablePersistence='enablePersistence' >

<e-layers>

<e-layer :shapeData='shapeData' ></e-layer>

</e-layers>

</ejs-maps>

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
return{
enablePersistence: true,
zoomSettings: {
enable: true,
},
shapeData: world_map,
}
},
provide: {
maps: [Zoom]
}
}
</script>
<style>
.wrapper {
max-width: 400px;
margin: 0 auto;
}
</style>
`
```

Accessibility in Vue Maps component

Maps has built-in accessibility features like screen reading, keyboard navigation, and WAI-ARIA attributes.

WAI-ARIA attributes

To meet accessibility standards, the Maps component follows to the WAI-ARIA patterns. In the Maps component, the following ARIA attributes are used:

Copyright © 2001 -2024 Syncfusion Inc.

| Attributes | Purpose |

| --- | --- |

| role=region | It specifies the Maps areas that do not support interactive functions like selection and highlight. |

| role=button | It specifies the Maps areas where interactive functions such as selection and highlight are available. |

| aria-label | Provides an accessible name for Maps elements such as geometric map shapes, legend title, data labels, and so on. To learn more, see the next topic. |

Screen reading in Maps

Accessibility in the Maps component ensures that all users, regardless of ability or disability, can use screen reading. The following Map elements will be read aloud using screen reading software, such as Narrator for Windows.

| Elements | Description |

| --- | --- |

| Shapes in the layer | Reads the names of the geographical shapes (such as countries, states, and regions) that appear on the Maps. |

| Legend title | Reads the contents of the legend's title as specified in Maps. |

| Legend item label | Reads the label of a legend item in Maps. |

| Data label | Reads the label specified for the shapes in the Maps layer. |

| Annotation | Reads the content specified in the annotation. |

| Marker template | Reads the content provided in the marker template. |

| Tooltip template | Reads the content provided in the tooltip template. |

| Data label template | Reads the content provided in the data label template. |

Keyboard Navigation

All the Maps actions can be controlled via keyboard keys. The applicable key combinations and their relative functionalities are listed below for the appropriate UI features available in the component.

Interaction Keys |Description

Tab |Moves to the next focusable element on the map, such as the legend or shape.

Shift + Tab |Moves to the previous focusable element on the map, such as the legend or shape.

+ |When zooming is enabled, zoom in operation can be performed.

- |When zooming is enabled, zoom out operation can be performed.

Left arrow |When zoomed in, the map can be scrolled to the left.

Right arrow |When zoomed in, the map can be scrolled to the right.

Up arrow |When zoomed in, the map can be scrolled upward.

Down arrow |When zoomed in, the map can be scrolled downward.

Copyright © 2001 -2024 Syncfusion Inc.

R |When zooming is enabled, reset operation can be performed.

Enter |The page can be navigated to the next and previous states in legend. Similarly, the selection can be made while navigating over the shape.

Ensuring accessibility

The Maps component's accessibility levels are ensured using an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Maps component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Maps component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/maps.html" %}

See also

- Accessibility in Syncfusion Vue components

Internationalization in Vue Maps component

Maps provide support for internationalization for the below elements.

- Data label
- Tooltip

For more information about number and date formatter, refer to the internationalization section.

<!-- markdownlint-disable MD036 -->

Globalization

Globalization is the process of designing and developing a component that works in different

cultures/locales. Internationalization library is used to globalize number, date, time values in

Maps component using format property in the Maps.

Numeric Format

The numeric formats such as currency, percentage and so on can be displayed in the tooltip and data labels of the Maps using the format property in the Maps. In the below example, the tooltip is globalized to **German** culture. When setting the useGroupingSeparator property as **true**, the numeric text in the Maps separates with the comma separator.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps  :format='format'
:useGroupingSeparator='useGroupingSeparator' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings'
:tooltipSettings='tooltipSettings'></e-layer>
                </e-layers>
            </ejs-maps>
```

```
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsTooltip } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
import { setCulture } from '@syncfusion/ej2-base';
setCulture('de');
Vue.use(MapsPlugin);
export default {
data () {
    return {
        format: 'c',
        useGroupingSeparator: true,
        shapeData: world_map,
        shapePropertyPath: 'name',
        shapeDataPath: 'Country',
        dataSource: [
            { "Country": "China", "Membership": "Permanent", population:
'38332521' },
            { "Country": "France", "Membership": "Permanent", population:
'19651127' },
            { "Country": "Russia", "Membership": "Permanent", population:
'3090416' },
            { "Country": "Kazakhstan", "Membership": "Non-Permanent",
population: '1232521' },
            { "Country": "Poland", "Membership": "Non-Permanent",
population: '90332521' },
            { "Country": "Sweden", "Membership": "Non-Permanent",
population: '383521' }
        ],
        shapeSettings: {
            colorValuePath: 'Membership',
            colorMapping: [
                {
                    value: 'Permanent', color: '#D84444'
                },
                {
                    value: 'Non-Permanent', color: '#316DB5'
                }
            ]
        },
        tooltipSettings: {
            visible: true,
            valuePath: 'population'
        }
    }
},
provide: {
    maps: [MapsTooltip]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
```

```
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs48" %}

Localization in Vue Maps component

The localization library allows localizing the default text content of the Maps component. The Maps component has the static text of some features such as tooltip of zoom toolbar, and that can be changed to any other culture(Arabic, Deutsch, French, etc) by defining the locale value and translation object.

<!-- markdownlint-disable MD033 -->

Locale key words	Text to display
Zoom	Zoom
ZoomIn	Zoom In
ZoomOut	Zoom Out
Reset	Reset
Pan	Pan

To load translation object in the application, use load function of **L10n** class. For more information about localization, refer here.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps :zoomSettings='zoomSettings' :locale='locale' >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapePropertyPath='shapePropertyPath' :shapeDataPath='shapeDataPath'
:dataSource='dataSource' :shapeSettings='shapeSettings' >
                    </e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
import { usMap } from './usa.js';
import { electionData } from './election-data.js';
import { L10n } from '@syncfusion/ej2-base';
Vue.use(MapsPlugin);
L10n.load({
    'ar-AR': {
        'maps': {
```

```
            ZoomIn: 'تكبير',
            ZoomOut: 'تصغير',
            Zoom: 'زوم',
            Pan: 'مقلاة',
            Reset: 'إعادة تعيين'
        },
    }
});
export default {
data () {
    return {
        zoomSettings: {
            enable: true
        },
        locale: 'ar-AR',
        shapeData: usMap,
        shapePropertyPath:  'name',
        shapeDataPath:  'State',
        dataSource : electionData,
        shapeSettings: {
        colorValuePath: 'Candidate',
            colorMapping: [
            {
                value: 'Trump', color: '#D84444'
            },
            {
                value: 'Clinton', color: '#316DB5'
            }]
        }
    }
},
provide: {
    maps: [Zoom]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs60" %}

Methods in Vue Maps component

Methods

This section explains the methods used in the Maps component.

getMinMaxLatitudeLongitude

The `getMinMaxLatitudeLongitude` method returns the minimum and maximum latitude and longitude values of the Maps visible area. This method returns a IMinMaxLatitudeLongitude object that contains the Maps minimum and maximum latitude and longitude coordinates.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div id="app">
          <div class='wrapper'>
             <ejs-button id='button' isToggle="true" v-
on:click.native='getMinMaxValues'> GetMinMaxLatitudeLongitude
             </ejs-button>
             <p id="coordinatesDisplay"></p>
            <ejs-maps id="maps" ref="maps" :zoomSettings='zoomSettings'
:centerPosition='centerPosition'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, Zoom} from '@syncfusion/ej2-vue-maps';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
Vue.use(ButtonPlugin);
export default {
data () {
    return{
        shapeData: world_map,
        zoomSettings:{
           enable:true,
           zoomFactor:7
        },
        centerPosition: {
            latitude: 21.815447,
            longitude: 80.1932,
        },
        markerSettings: [
          {
               visible:true,
                height:25,
                width:25,
                shape:"Circle",
                animationDuration:1500,
                dataSource:[
                  {
                    latitude: 22.572646,
                    longitude: 88.363895,
                  },
                  {
                    latitude: 25.0700428,
                    longitude: 67.2847875,
                  }
                ]
          }
        ]
    }
},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
provide: {
    maps: [Marker, Zoom]
},
methods: {
  formatKey: function(key) {
    if (key === 'minLatitude') {
      return 'Minimum Latitude';
    } else if (key === 'maxLatitude') {
      return 'Maximum Latitude';
    } else if (key === 'minLongitude') {
      return 'Minimum Longitude';
    } else if (key === 'maxLongitude') {
      return 'Maximum Longitude';
    }
  },
  getMinMaxValues: function() {
    var mapBoundCoordinates;
    mapBoundCoordinates =
this.$refs.maps.ej2Instances.getMinMaxLatitudeLongitude();
    const displayDiv = document.getElementById('coordinatesDisplay');
    displayDiv.innerHTML = '';
    if (mapBoundCoordinates) {
      for (const key in mapBoundCoordinates) {
        if (Object.hasOwnProperty.call(mapBoundCoordinates, key)) {
          const p = document.createElement('p');
          const formattedKey = this.formatKey(key);
          p.textContent = `${formattedKey}: ${mapBoundCoordinates[key]}`;
          displayDiv.appendChild(p);
        }
      }
    } else {
      displayDiv.textContent = 'No coordinates available';
    }
  }
}
}
</script>
<style>
  .wrapper {
    max-width: 450px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs107" %}

How To

Annotation in Vue Maps component

Annotations are used to mark the specific area of interest in the Maps with texts, shapes, or images. Any number of annotations can be added to the Maps component.

Initialize the Maps component with annotation option, text content or ID of an HTML element or an HTML string can be specified to render a new element that needs to be displayed in the Maps by using the content property. To specify the content position with x and y properties as mentioned in the following example.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps :annotations='annotations'>
                <e-layers>
                    <e-layer :shapeData='shapeData'
:shapeSettings='shapeSettings' >
                    </e-layer>
                </e-layers>
            </ejs-maps>
    <svg height="150" width="400">
        <defs>
            <linearGradient id="grad1" x1="0%" y1="0%" x2="0%" y2="100%">
                <stop offset="0%" style="stop-color:#C5494B;stop-
opacity:1"></stop>
                <stop offset="100%" style="stop-color:#4C134F;stop-
opacity:1"></stop>
            </linearGradient>
        </defs>
    </svg>
    <div id="maps-annotation" style="display: none;">
        <div id="annotation">
            <div>
                <p style="margin-left:10px;font-size:13px;font-
weight:500">Facts about Africa</p>
            </div>
            <hr style="margin-top:-3px;margin-bottom:10px;border:0.5px solid
#DDDDDD">
            <div>
                <ul style="list-style-type:disc; margin-left:-20px;margin-
bottom:2px; font-weight:400">
                    <li>Africa is the second largest and second most
populated continent in the world.</li>
                    <li style="padding-top:5px;">Africa has 54 sovereign
states and 10 non-sovereign territories.</li>
                    <li style="padding-top:5px;">Algeria is the largest
country in Africa, where as Mayotte is the smallest.</li>
                </ul>
            </div>
        </div>
    </div>
        <div id="compass-maps" style="display: none;">
          <img
src="https://ej2.syncfusion.com/vue/demos/src/maps/images/compass.svg"
height="75px" width="75px">
        </div>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent, Annotations } from '@syncfusion/ej2-vue-
maps';
import { africa } from './africa.js';
Vue.use(MapsPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
data () {
    return {
        shapeData: africa,
        shapeSettings: {
            fill: 'url(#grad1)'
        },
        annotations:[{
            content:'#maps-annotation', // To insert the text content
            x: '0%', y: '70%'
        },
        {
            content: '#compass-maps',  // To insert the image
            x: '80%', y: '5%'
        }]
    }
},
provide: {
    maps: [ Annotations ]
}
}
</script>
<style>
    .wrapper {
        max-width: 400px;
        margin: 0 auto;
    }
    #annotation {
        color: #DDDDDD;
        font-size: 12px;
        font-family: Roboto;
        background: #3E464C;
        margin: 20px;
        padding: 10px;
        border-radius: 2px;
        width: 300px;
        box-shadow: 0px 2px 5px #666;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs41" %}

Custom path in Vue Maps component

Maps component can be customized as the desired layout using the custom path map feature. Here, the Maps component has been showcased with normal geometry type shapes to represent the bus seat selection layout.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
        <div style="border: 3px solid darkgray; width: 200; display;
'block'; margin: auto;">
            <ejs-maps :height='height'>
                <e-layers>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <e-layer :shapeData='shapeData'
:geometryType='geometryType'>
                    </e-layer>
                </e-layers>
            </ejs-maps>
            </div>
      </div>
</div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, MapsComponent } from '@syncfusion/ej2-vue-maps';
import { seat } from './seat.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
      geometryType: 'Normal',
      shapeData: seat,
      height: '400'
    }
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs42" %}

Drilldown in Vue Maps component

By clicking a continent, all the countries available in that continent can be viewed using the drill-down feature. For example, the countries in the Africa continent have been showcased here. To showcase all the countries in Africa continent by clicking the shapeSelected event as mentioned in the following example.

APP.VUE

```
<template>
    <div id="app">
        <div class='wrapper'>
            <ejs-maps  ref="maps" :layers='layers'
:shapeSelected='shapeSelected'>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker } from '@syncfusion/ej2-vue-maps';
import { default_data } from './default-data.js';
import { world_map } from './world-map.js';
import { africa } from './africa.js';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(MapsPlugin);
export default {
data () {
    return {
        layers: [
        {
            layerType: 'Geometry',
            shapeData: world_map,
            shapePropertyPath: 'continent',
            shapeDataPath: 'continent',
            dataSource:default_data,
            shapeSettings: {
                colorValuePath: 'drillColor'
            },
            markerSettings: [{
                visible: true,
                template: '<div id="marker3" class="markerTemplate">Africa'
+
                    '</div>',
                dataSource: [
                    { latitude: 10.97274101999902, longitude: 16.390625 }
                ],
                animationDuration: 0
            }]
        },
        {
            layerType: 'Geometry',
            shapeData: africa,
            shapeSettings: {
                fill: '#80306A'
            },
            highlightSettings: {
                enable: true,
                fill: '#80306A'
            }
        }]
    }
},
provide: {
    maps: [Marker]
},
methods: {
    shapeSelected: function(args){
        let shape = (args.shapeData).continent;
        if (shape === 'Africa') {
            this.$refs.maps.ej2Instances.baseLayerIndex = 1;
            this.$refs.maps.ej2Instances.refresh();
        }
    }
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs43" %}

Marker type in Vue Maps component

Add different types of markers

Different marker objects can be added to the Maps component using the marker settings. To update different marker settings in Maps, please follow the given steps:

Step 1:

Initialize the Maps component with marker settings. Here, a marker has been added with specified latitude and longitude of California by using the dataSource property. To customize the shape of the marker using the shape property and change the border color and width of the marker using the border property as mentioned in the following example.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: world_map,
        markerSettings: [{
        dataSource: [
            { latitude: 40.7424509, longitude: -74.0081468, city: 'New York'
}],
        visible:true,
        shape: 'Circle',
        fill: 'white',
        width: 3,
        animationDuration:0,
            border: { width: 2, color: '#333' }
        }]
    }
},
provide: {
    maps: [Marker]
}
}
```

```
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs44" %}

Step 2:

Customize the above option for n number of markers as mentioned in the following example.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps >
                <e-layers>
                    <e-layer :shapeData='shapeData'
:markerSettings='markerSettings'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Marker, } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        shapeData: world_map,
        markerSettings: [{
            dataSource: [
                { latitude: 37.6276571, longitude: -122.4276688, city: 'San
Bruno' },
            ],
            visible:true,
            shape:'Circle',
            fill:'white',
            width:3,
            animationDuration:0,
            border:{ width: 2, color: '#333'}
        },
        {
            dataSource: [
                { latitude: 33.5302186, longitude: -117.7418381, city:
'Laguna Niguel' },
            ],
            visible:true,
            shape:'Rectangle',
            fill:'yellow',
```

```
            width:15,
            height:4,
            animationDuration:0,
            border: { width: 2, color: '#333' }
        },
        {
            dataSource: [
                { latitude: 40.7424509, longitude: -74.0081468, city: 'New
York' }
            ],
            visible:true,
            shape:'Diamond',
            fill:'white',
            width:10,
            height:10,
            animationDuration:0,
            border: { width: 2, color: 'blue' }
        }]
    }
},
provide: {
    maps: [Marker]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs45" %}

Multiple layer in Vue Maps component

The multilayer support allows loading multiple shape files in a single container and enables Maps to display more information. The shape layer is the main layer of the Maps. Multiple layers can be added in a shape layer as **SubLayer** using the type property.

APP.VUE

```
<template>
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :layers='layers'>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin } from '@syncfusion/ej2-vue-maps';
import { usMap } from './usa.js';
import { texas } from './texas.js';
import { california } from './california.js';
Vue.use(MapsPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
data () {
    return {
       layers: [{
            shapeData: usMap,
            shapeSettings: {
                fill: '#E5E5E5',
                border: {
                    color: 'black',
                    width: 0.1
                }
            }
        },
        {
            shapeData: texas,
            type: 'SubLayer',
            shapeSettings: {
                fill: 'rgba(141, 206, 255, 0.6)',
                border: {
                    color: '#1a9cff',
                    width: 0.25
                }
            }
        },
        {
            shapeData: california,
            type: 'SubLayer',
            shapeSettings: {
                fill: 'rgba(141, 206, 255, 0.6)',
                border: {
                    color: '#1a9cff',
                    width: 0.25
                }
            }
        }
    ]}
}
}
</script>
<style>
    .wrapper {
        max-width: 400px;
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs46" %}

Zooming in Vue Maps component

The center position zooming can be achieved by using the centerPosition and zoomFactor properties as mentioned in the following example. The center position is used to configure the zoom level of Maps, and the zoom factor is used to specify the center position where the Maps should be displayed.

APP.VUE

```
<template>
```

```
    <div id="app">
          <div class='wrapper'>
            <ejs-maps :zoomSettings='zoomSettings'
:centerPosition='centerPosition'>
                <e-layers>
                    <e-layer :shapeData='shapeData'></e-layer>
                </e-layers>
            </ejs-maps>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MapsPlugin, Zoom } from '@syncfusion/ej2-vue-maps';
import { world_map } from './world-map.js';
Vue.use(MapsPlugin);
export default {
data () {
    return {
        zoomSettings: {
            enable: true,
            zoomFactor: 13
        },
        centerPosition: {
            latitude: 25.54244147012483,
            longitude: -89.62646484375
        },
        shapeData: world_map,
    }
},
provide: {
    maps: [Zoom]
}
}
</script>
<style>
  .wrapper {
    max-width: 400px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/maps/getting-started-cs47" %}

MaskedTextBox

Getting Started with the Vue Maskedtextbox Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Maskedtextbox component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Copyright © 2001 -2024 Syncfusion Inc.

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

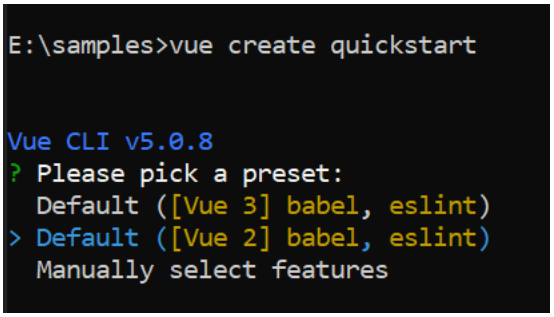
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Maskedtextbox component as an example. Install the @syncfusion/ej2-vue-inputs package by running the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Maskedtextbox component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

If you want to refer the combined component styles, please make use of our CRG (Custom Resource Generator) in your application.

Add Syncfusion Vue component

Follow the below steps to add the Vue Maskedtextbox component using Composition API or Options API:

1\. First, import and register the Maskedtextbox component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { MaskedTextBoxComponent as EjsMaskedtextbox } from "@syncfusion/ej2-vue-inputs";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { MaskedTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
export default {
components: {
'ejs-maskedtextbox': MaskedTextBoxComponent
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

Adding MaskedTextBox component

Add the EJ2 Vue MaskedTextBox using <ejs-maskedtextbox> to the <template> section of the App.vue file in src directory, the content attribute of the MaskedTextBox component is provided as name in data option in the <script> section.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<div class="control-label">MaskedTextbox
</div>
<ejs-maskedtextbox></ejs-maskedtextbox>
</div>
</template>
<script setup>
import { MaskedTextBoxComponent as EjsMaskedtextbox } from "@syncfusion/ej2-vue-inputs";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<div class="control-label">MaskedTextbox
</div>
<ejs-maskedtextbox></ejs-maskedtextbox>
</div>
</template>
<script>
import { MaskedTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
export default {
components: {
'ejs-maskedtextbox': MaskedTextBoxComponent
},
data () {
return {}
}
}
</script>
```

Set the mask

You can set the mask to the MaskedTextBox to validate the user input by using the mask property. To know more about the usage of mask and configuration, refer to this link.

The following example demonstrates the usage of mask element 0 that allows any single digit from 0 to 9.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<div class="control-label">MaskedTextBox
</div>
<ejs-maskedtextbox mask="000-000-0000"></ejs-maskedtextbox>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script setup>
import { MaskedTextBoxComponent as EjsMaskedtextbox } from "@syncfusion/ej2-vue-inputs";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<div class="control-label">MaskedTextBox
</div>
<ejs-maskedtextbox mask="000-000-0000"></ejs-maskedtextbox>
</div>
</template>
<script>
import { MaskedTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
export default {
components: {
'ejs-maskedtextbox': MaskedTextBoxComponent
}
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-maskedtextbox mask='000-000-0000' placeholder='MaskedTextBox' floatLabelType='Always'></ejs-maskedtextbox>
    </div>
  </div>
</template>
<script setup>
import { MaskedTextBoxComponent as EjsMaskedtextbox } from "@syncfusion/ej2-vue-inputs";
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div class='wrap'>
```

```
        <ejs-maskedtextbox mask='000-000-0000' placeholder='MaskedTextBox'
floatLabelType='Always'></ejs-maskedtextbox>
    </div>
  </div>
</template>
<script>
import { MaskedTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
export default {
  components: {
    'ejs-maskedtextbox': MaskedTextBoxComponent
  },
  data () {
    return {}
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/masked-textbox/getting-started-cs1" %}

Model binding

In MaskedTextBox, the value property supports model binding functionality.

The following example demonstrates model binding functionality with the MaskedTextBox and HTML input element.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
  <div class='wrap'>
        <div class='e-input-group' style='margin-bottom:30px'>
        <input class='e-input' type='text' v-model='value'
placeholder='Mobile Number'  />
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </div>
        <div>
           <ejs-maskedtextbox v-model="value" :value="value" mask='000-000-
0000'></ejs-maskedtextbox>
        </div>
    </div>
  </div>
</template>
<script setup>
import { MaskedTextBoxComponent as EjsMaskedtextbox } from "@syncfusion/ej2-
vue-inputs";
const value = '';

</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
  margin: 35px auto;
  width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
  <div class='wrap'>
        <div class='e-input-group' style='margin-bottom:30px'>
        <input class='e-input' type='text' v-model='value'
placeholder='Mobile Number'  />
        </div>
        <div>
           <ejs-maskedtextbox v-model="value" :value="value" mask='000-000-
0000'></ejs-maskedtextbox>
        </div>
    </div>
  </div>
</template>
<script>
import { MaskedTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
export default {
  components: {
    'ejs-maskedtextbox': MaskedTextBoxComponent
  },
  data () {
    return {
      value : ''
    }
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  margin: 35px auto;
  width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/masked-textbox/model-cs1" %}

See Also

- How to perform custom validation using FormValidator
- How to customize the UI appearance of the control
- How to set cursor position while focus on the input textbox
- How to display numeric keypad when focus on mobile devices

Getting Started with the Vue MaskedTextBox Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue MaskedTextBox component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

Copyright © 2001 -2024 Syncfusion Inc.

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Copyright © 2001 -2024 Syncfusion Inc.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue MaskedTextBox component as an example. To use the Vue MaskedTextBox component in the project, the @syncfusion/ej2-vue-inputs package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the MaskedTextBox component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue MaskedTextBox component using Composition API or Options API:

1.First, import and register the MaskedTextBox component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { MaskedTextBoxComponent as EjsMaskedtextbox } from "@syncfusion/ej2-vue-inputs";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import { MaskedTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: "App",
components: {
'ejs-maskedtextbox': MaskedTextBoxComponent,
}
}
</script>
```

2.In the template section, define the MaskedTextBox component with the dataSource property and column definitions.

~/SRC/APP.VUE

```
<template>
<div class="control_wrapper">
<ejs-maskedtextbox></ejs-maskedtextbox>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-maskedtextbox></ejs-maskedtextbox>
</div>
</template>
<script setup>
import { MaskedTextBoxComponent as EjsMaskedtextbox } from "@syncfusion/ej2-vue-inputs";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.control_wrapper {
margin: 0 auto;
width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-maskedtextbox></ejs-maskedtextbox>
</div>
</template>
<script>
import { MaskedTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
name: 'App',
components: {
"ejs-maskedtextbox": MaskedTextBoxComponent
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.control_wrapper {
margin: 0 auto;
width: 240px;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

Set the mask

You can set the mask to the MaskedTextBox to validate the user input by using the

mask property. To know more about

the usage of mask and configuration, refer to this link.

The following example demonstrates the usage of mask element 0 that allows any single digit from 0 to 9.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrap'>
<ejs-maskedtextbox mask='000-000-0000' placeholder='MaskedTextBox'
floatLabelType='Always'></ejs-maskedtextbox>
</div>
</div>
</template>
<script setup>
import { MaskedTextBoxComponent as EjsMaskedtextbox } from "@syncfusion/ej2-
vue-inputs";
const data = [];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
margin: 0 auto;
width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrap'>
<ejs-maskedtextbox mask='000-000-0000' placeholder='MaskedTextBox'
floatLabelType='Always'></ejs-maskedtextbox>
</div>
</div>
</template>
<script>
import { MaskedTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: 'App',
components: {
"ejs-maskedtextbox": MaskedTextBoxComponent
},
data () {
return {}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
margin: 0 auto;
width: 240px;
}
</style>
```

The output will appear as follows:

Model binding

In MaskedTextBox, the value property supports model binding functionality.

The following example demonstrates model binding functionality with the MaskedTextBox and HTML input element.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrap'>
```

```
<div class='e-input-group' style='margin-bottom:30px'>
<input class='e-input' type='text' v-model='value' placeholder='Mobile
Number' />
</div>
<ejs-maskedtextbox v-model="value" :value="value" mask='000-000-0000'></ejs-
maskedtextbox>
</div>
</div>
</template>
<script setup>
import { ref } from 'vue';
import { MaskedTextBoxComponent as EjsMaskedtextbox } from "@syncfusion/ej2-
vue-inputs";
const value = ref('');
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
margin: 35px auto;
width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrap'>
<div class='e-input-group' style='margin-bottom:30px'>
<input class='e-input' type='text' v-model='value' placeholder='Mobile
Number' />
</div>
<ejs-maskedtextbox v-model="value" :value="value" mask='000-000-0000'></ejs-
maskedtextbox>
</div>
</div>
</template>
<script>
import { MaskedTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: 'App',
components: {
"ejs-maskedtextbox": MaskedTextBoxComponent
},
data () {
return {
value : ''
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.wrap {
margin: 35px auto;
width: 240px;
}
</style>
```

The output will appear as follows:

See Also

- How to perform custom validation using FormValidator
- How to customize the UI appearance of the control
- How to set cursor position while focus on the input textbox
- How to display numeric keypad when focus on mobile devices

Mask configuration in Vue Maskedtextbox component

The mask is a combination of standard and custom mask elements, that validates the user input based on its behavior.

When the mask value is empty, the MaskedTextBox behaves as an input element with text type.

Standard mask elements

The following table shows the list of mask elements and its behavior based on

MSDN standard.

The mask can be formed by combining any one or more of these mask elements.

| Mask Element | Description |

| ---------- | ---------- |

| 0 | Digit required. This element will accept any single digit from **0** to **9**. |

| 9 | Digit or space, optional. |

| # | Digit or space, optional, Plus(**+**) and minus(**-**) signs are allowed. |

| L | Letter required. It will accept letters **a-z** and **A-Z**. |

| ? | Letter or space, optional. |

| & | Requires a character. |

| C | Character or space, optional. |

| A | Alphanumeric **(A-Za-z0-9)** required.|

| a | Alphanumeric **(A-Za-z0-9)** or space, optional. |

| < | Shift down. Converts all characters to lower case. |

Copyright © 2001 -2024 Syncfusion Inc.

| > | Shift up. Converts all characters to upper case. |

| | | Disable a previous shift up or shift down. |

| \\\\ | Escapes a mask character, turning it into a literal. |

| All other characters | Literals. All non-mask elements (literals) will appear as themselves within MaskedTextBox. |

The following example demonstrates the usage of standard mask elements.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-maskedtextbox mask='#####' placeholder='Mask ##### (ex:
012+-)' floatLabelType='Always'></ejs-maskedtextbox>
        </div>
        <div class='wrap'>
            <ejs-maskedtextbox mask='LLLLLL' placeholder='Mask LLLLLL (ex:
Sample)' floatLabelType='Always'></ejs-maskedtextbox>
        </div>
        <div class='wrap'>
            <ejs-maskedtextbox mask='&&&&&' placeholder='Mask &&&&& (ex:
A12@#)' floatLabelType='Always'></ejs-maskedtextbox>
        </div>
        <div class='wrap'>
            <ejs-maskedtextbox mask='>LLL<LLL' placeholder='Mask >LLL<LL
(ex: SAMple)' floatLabelType='Always'></ejs-maskedtextbox>
        </div>
        <div class='wrap'>
            <ejs-maskedtextbox mask='\\A999' placeholder='Mask \\A999 (ex:
A321)' floatLabelType='Always'></ejs-maskedtextbox>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { MaskedTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(MaskedTextBoxPlugin);
export default {
  data () {
    return {}
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
  margin: 20px auto;
  width: 240px;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/masked-textbox/mask-configuration/standard-cs1" %}

Custom mask elements

Other than the above standard mask elements, the mask can be configured with the custom characters or regular expression to define a custom behavior.

Custom characters

You can define any of the non-mask element as the mask element and its behavior through the customCharacters property.

In the following example, non-mask element P accepts the values P, A, p, a and M accepts the values M, m as mentioned in the custom characters collection.

APP.VUE

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-maskedtextbox :customCharacters='customCharacters' mask='00:00
>PM' placeholder='Time (ex: 10:00 PM, 10:00 AM)'
floatLabelType='Always'></ejs-maskedtextbox>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { MaskedTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(MaskedTextBoxPlugin);
export default {
  data () {
    return {
        customCharacters: {
            P: 'P,A,a,p',
            M: 'M,m'
        }
    }
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/masked-textbox/mask-configuration/custom-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Regular expression

Instead of the mask element, you can define your own regular expression to validate the input of a particular input place. The regular expressions should be wrapped by the square brackets (e.g.,: [Regex]).

In the following example, regular expression has been set for each input places.

APP.VUE

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-maskedtextbox mask='[0-2][0-9][0-9].[0-2][0-9][0-9].[0-2][0-
9][0-9].[0-2][0-9][0-9]' placeholder='IP Address (ex: 212.212.111.222)'
id='mask1' name='mask_value' floatLabelType='Always'></ejs-maskedtextbox>
    </div>
  </div>
  </template>
<script>
import Vue from "vue";
import { MaskedTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(MaskedTextBoxPlugin);
export default {
  data () {
    return {}
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/masked-textbox/mask-configuration/regular-cs1" %}

Prompt character

The Prompt character is a prompting symbol in the MaskedTextBox for the mask elements. The symbol is used to show the input positions in the MaskedTextBox. You can customize the prompt character of MaskedTextBox by using the promptChar property.

The following example demonstrates the MaskedTextBox with customized prompt character as *.

APP.VUE

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-maskedtextbox mask='999-999-9999' promptChar="#"></ejs-
maskedtextbox>
    </div>
  </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { MaskedTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(MaskedTextBoxPlugin);
export default {
  data () {
    return {}
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/masked-textbox/mask-configuration/prompt-cs1" %}

Accessibility in Vue Maskedtextbox component

The Maskedtextbox component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Maskedtextbox component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The MaskedTextBox is characterized with complete ARIA Accessibility support that helps to access through the on-screen readers and other assistive technology devices. This component is designed with the reference of the guidelines document given in WAI ARAI Accessibility practices.

The MaskedTextBox uses the `textbox` role and following ARIA properties for its element based on its state.

| **Property** | **Functionality** |

| --- | --- |

| aria-live | The `aria-live` attribute indicates the priority of updates to a live region. |

| aria-disabled | The `aria-disabled` property indicates the disabled state of the MaskedTextBox. |

| aria-valuenow | The `aria-valuenow` property specifies the current value of the MaskedTextBox. |

| aria-invalid | The `aria-invalid` property indicates that the user input is incorrect or not within the acceptable ranges. |

| aria-placeholder | The `aria-placeholder` is a short hint to help the users with data entry when the MaskedTextBox has no value. |

| aria-labelledby | The `aria-labelledby` property indicates the floating label element of the MaskedTextBox. |

Ensuring accessibility

The MaskedTextBox component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the MaskedTextBox component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the MaskedTextBox component with accessibility tools.

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "https://ej2.syncfusion.com/accessibility/masked-textbox.html" %}

See also

- Accessibility in Syncfusion Vue components

Two way binding in Vue Maskedtextbox component

Two-way binding can be achieved by using the v-model directive in Vue. In the following sample, when you change the value in one MaskedTextBox component, v-model will automatically update the value in the other MaskedTextBox.

The following example demonstrates how to set the two-way-binding in the MaskedTextBox.

APP.VUE

```
<template>
<div id="app">
    <div id="wrapper1">
        <ejs-maskedtextbox floatLabelType="Auto" placeholder="Enter a value"
mask="999-999-9999" v-model="value"></ejs-textbox>
    </div>
    <div id="wrapper2">
        <ejs-maskedtextbox floatLabelType="Auto" placeholder="Enter a value"
mask="999-999-9999" v-model="value"></ejs-textbox>
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { MaskedTextBoxPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(MaskedTextBoxPlugin);
export default {
 data: function(){
        return {
            value: null
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#wrapper1{
    min-width: 250px;
    float: left;
    margin-left: 100px;
}
#wrapper2{
    min-width: 250px;
    float: right;
    margin-right: 100px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/masked-textbox/two-way-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Style appearance in Vue Maskedtextbox component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of MaskedTextBox wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
/* To specify height, font size, and border */

.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input, .e-input-group textarea.e-input, .e-input-group.e-control-wrapper textarea.e-input {

font-size: 20px;

border-color: red;

height: 40px;

border: 2px solid;

}
```

Customizing the MaskedTextBox element on hovering

Use the following CSS to customize the Input Mask element on hovering

```
/* To specify border */

.e-input-group input.e-input, .e-input-group input.e-input:hover:not(.e-success):not(.e-warning):not(.e-error):not([disabled]):not(:focus), .e-input-group.e-control-wrapper input.e-input,.e-input-group.e-control-wrapper input.e-input:hover:not(.e-success):not(.e-warning):not(.e-error):not([disabled]):no(:focus){

border: 3px solid red;

}
```

How To

Perform custom validation using form validator in Vue Maskedtextbox component

To perform custom validation on the MaskedTextBox use the FormValidator along with custom validation rules.

In the following example, the MaskedTextBox is validated for invalid mobile number by adding custom validation in the rules collection of the FormValidator.

APP.VUE

```
<template>
  <div id="app">
    <div class='wrap'>
      <form id="form-element" class="form-horizontal">
        <div class="form-group">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          <ejs-maskedtextbox id="mask" type="text" name="mask" class="form-
control" mask='000-000-0000' placeholder='Mobile Number'
floatLabelType='Always'></ejs-maskedtextbox>
          <button type="button" id="submit_btn" style="margin-top: 10px" v-
on:click="onClick">Submit</button>
        </div>
      </form>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { MaskedTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
Vue.use(MaskedTextBoxPlugin);
export default {
  data () {
    return {}
  },
  mounted: function() {
    var customFn = function (args) {
        if (args.element.ej2_instances[0].value.length !== 0) {
          return args.element.ej2_instances[0].value.length >= 10; }
          else {
            return true;
        }
      };
      var custom = function (args) {
        if (args.element.ej2_instances[0].value.length === 0) {
          return 0;
        }
        else {
          return args.element.ej2_instances[0].value.length;
        }
          //return args.element.ej2_instances[0].value.length === 0;
      };
      // sets required property in the FormValidator rules collection
      var options = {
          rules: {
              'mask': { numberValue: [customFn, 'Enter valid mobile number']
},
          },
      };
      // defines FormValidator to validate the MaskedTextBox
       this.formObject = new FormValidator('#form-element', options);
      this.formObject.addRules('mask', { maxLength: [custom, 'Enter mobile
number'] });
      // places error label outside the MaskedTextBox using the
customPlacement event of FormValidator
      this.formObject.customPlacement = function (element, errorElement) {
document.querySelector(".form-group").appendChild(errorElement);
};
  },
  methods: {
    onClick: function (event) {
      this.formObject.validate("mask");
      var ele = document.getElementById('mask');
```

```
      // checks for incomplete value and alerts the format submit
      if (ele.value !== "" &&
ele.value.indexOf(mask.ej2_instances[0].promptChar) === -1) {
            alert("Submitted");
      }
    }
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
.e-mask.e-control-wrapper {
  margin-bottom: 20px;
}
label.e-error {
  margin-top: -42px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/masked-textbox/how-to/validation-cs1" %}

Set cursor position while focus on the input textbox in Vue Maskedtextbox component

By default, on focusing the MaskedTextBox the entire mask gets selected. You can customize by using any one of the following methods:

- Setting cursor position at the start of the MaskedTextBox.
- Setting cursor position at the end of the MaskedTextBox.
- Setting cursor at the specified position in the MaskedTextBox.

The **selectionStart** and **selectionEnd** set to **0** instead of the input element value's length, when we focus on a MaskedTextBox control filled with all mask characters. This is the default behavior of the HTML 5 input element.

Following is an example that demonstrates the above cases to set cursor position in the MaskedTextBox using focus event.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-maskedtextbox mask='00000-00000' value='93828-32132'
placeholder='Default cursor position' floatLabelType='Always'></ejs-
maskedtextbox>
        </div>
        <div class='wrap'>
            <ejs-maskedtextbox mask='00000-00000' value='83929-4342'
placeholder='Cursor positioned at start' floatLabelType='Always'
:focus='focus'></ejs-maskedtextbox>
```

```
        </div>
        <div class='wrap'>
            <ejs-maskedtextbox mask='00000-00000' value='83929-3213'
placeholder='Cursor positioned at end' floatLabelType='Always'
:focus='endfocus'></ejs-maskedtextbox>
        </div>
        <div class='wrap'>
            <ejs-maskedtextbox mask='+1 000-000-0000' value='234-432-432'
placeholder='Cursor at specified position' floatLabelType='Always'
:focus='specifiedfocus'></ejs-maskedtextbox>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { MaskedTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(MaskedTextBoxPlugin);
export default {
  data () {
    return {}
  },
  methods: {
      focus: function(args) {
           args.selectionEnd= args.selectionStart = 0;
      } ,
      endfocus: function(args){
          args.selectionStart=args.selectionEnd = args.maskedValue.length;
      },
      specifiedfocus: function(args){
          args.selectionStart = 3;
          args.selectionEnd = 3;
      }
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
  margin: 20px auto;
  width: 240px;
}
.e-widget {
  padding-bottom: 12px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/masked-textbox/how-to/cursor-cs1" %}

Display numeric keypad when focus on mobile devices in Vue Maskedtextbox component

By default, on focusing the MaskedTextBox, alphanumeric keypad will be displayed on mobile devices. Sometimes only numeric keypad for number

values is needed, and this can be achieved by setting "type" property to tel. Refer to the following example to enable numeric keypad in MaskedTextBox.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-maskedtextbox type='tel' class='form-control' mask='999-99999'
value='342-45432'></ejs-maskedtextbox>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { MaskedTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(MaskedTextBoxPlugin);
export default {
  data () {
    return {}
  },
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/masked-textbox/how-to/numeric-cs1" %}

Customize the ui appearance of the control in Vue Maskedtextbox component

The appearance of the MaskedTextBox can be changed by adding custom cssClass to the component and enabling styles. Refer to the following example to change the appearance of the MaskedTextBox.

APP.VUE

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-maskedtextbox mask='00000' value='42648' placeholder='Enter
User ID' cssClass='e-style' floatLabelType='Always' :focus='focus'></ejs-
maskedtextbox>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { MaskedTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(MaskedTextBoxPlugin);
export default {
  data () {
    return {}
  },
  methods: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      focus: function(args) {
           args.selectionEnd = args.selectionStart;
           }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
    margin: 0 auto;
    width: 240px;
}
.e-widget.e-mask.e-style .e-control.e-maskedtextbox {
  color: #00fff ;
  letter-spacing: 10px ;
  font-size: xx-large ;
  border: 1px ;
  border-color: #ffffff ;
}
.e-widget.e-control-wrapper.e-mask.e-float-input.e-style .e-float-
line::before {
  background: #ffffff ;
}
.e-widget.e-control-wrapper.e-mask.e-float-input.e-style .e-float-
line::after {
  background: #ffffff ;
}
.e-widget.e-control-wrapper.e-mask.e-float-input.e-style .e-float-text.e-
label-top {
  color: #00fff ;
  font-size: medium ;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/masked-textbox/how-to/cssClass-cs1" %}

Mention

Getting Started with the Vue Mention Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Mention component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies are required to use the Mention component in your application.

`js

|-- @syncfusion/ej2-vue-dropdowns

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-dropdowns

Copyright © 2001 -2024 Syncfusion Inc.

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-data

|-- @syncfusion/ej2-lists

|-- @syncfusion/ej2-popups

|-- @syncfusion/ej2-buttons

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

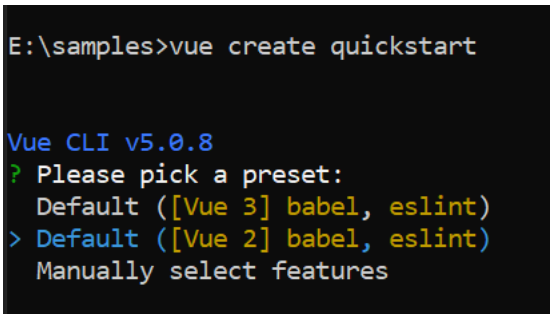
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Mention component as an example. Install the @syncfusion/ej2-vue-dropdowns package by running the following command:

`bash

npm install @syncfusion/ej2-vue-dropdowns --save

`

or

`bash

yarn add @syncfusion/ej2-vue-dropdowns

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Mention component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Mention component using Composition API or Options API:

1\. First, import and register the Mention component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { MentionComponent as EjsMention } from "@syncfusion/ej2-vue-
dropdowns";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { MentionComponent } from "@syncfusion/ej2-vue-dropdowns";
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
components: {
'ejs-mention': MentionComponent
}
}
</script>
```

Adding Mention component

Add the EJ2 Vue Mention using <ejs-mention> to the <template> section of the App.vue file in src directory. To use the Mention component properly, the target property should be configured so that it renders the Mention component in the configured element.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<label id="comment" >Comments</label>
<div id="mentionElement" placeholder = "Type @ and tag user"></div>
<ejs-mention id='defaultMention' :target='mentionTarget'></ejs-mention>
</div>
</template>
<script setup>
import { MentionComponent as EjsMention } from "@syncfusion/ej2-vue-
dropdowns";
const mentionTarget = "#mentionElement";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
#app {
color: #008cff;
height: 40px;
left: 15%;
position: absolute;
top: 10%;
width: 30%;
}
#comment {
font-size: 15px;
font-weight: 600;
}
#mentionElement {
min-height: 100px;
border: 1px solid #D7D7D7;
border-radius: 4px;
padding: 8px;
font-size: 14px;
width: 600px;
}
div#mentionElement[placeholder]:empty:before {
content: attr(placeholder);
color: #555;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<label id="comment" >Comments</label>
<div id="mentionElement" placeholder = "Type @ and tag user"></div>
<ejs-mention id='defaultMention' :target='mentionTarget'></ejs-mention>
</div>
</template>
<script>
import { MentionComponent } from "@syncfusion/ej2-vue-dropdowns";
export default {
components: {
'ejs-mention': MentionComponent
},
name: 'app',
data: function() {
return {
mentionTarget: "#mentionElement",
};
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
#app {
color: #008cff;
height: 40px;
left: 15%;
position: absolute;
top: 10%;
width: 30%;
}
#comment {
font-size: 15px;
font-weight: 600;
}
#mentionElement {
min-height: 100px;
border: 1px solid #D7D7D7;
border-radius: 4px;
padding: 8px;
font-size: 14px;
width: 600px;
}
div#mentionElement[placeholder]:empty:before {
content: attr(placeholder);
color: #555;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

Binding data source

After initialization, populate the Mention with data using the dataSource property. Here, an array of string values is passed to the Mention component.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<label id="comment" >Comments</label>
<div id="mentionElement" placeholder = "Type @ and tag user"></div>
<ejs-mention id='defaultMention' :target='mentionTarget'
:dataSource='userData'></ejs-mention>
</div>
</template>
<script setup>
import { MentionComponent as EjsMention } from "@syncfusion/ej2-vue-
dropdowns";
const mentionTarget = "#mentionElement";
const userData = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
#app {
color: #008cff;
height: 40px;
left: 15%;
position: absolute;
top: 10%;
width: 30%;
}
#comment {
font-size: 15px;
font-weight: 600;
}
#mentionElement {
min-height: 100px;
border: 1px solid #D7D7D7;
border-radius: 4px;
padding: 8px;
font-size: 14px;
width: 600px;
}
div#mentionElement[placeholder]:empty:before {
content: attr(placeholder);
color: #555;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<label id="comment" >Comments</label>
<div id="mentionElement" placeholder = "Type @ and tag user"></div>
<ejs-mention id='defaultMention' :target='mentionTarget'
:dataSource='userData'></ejs-mention>
</div>
</template>
<script>
import { MentionComponent } from "@syncfusion/ej2-vue-dropdowns";
export default {
components: {
'ejs-mention': MentionComponent
},
name: 'app',
data: function() {
return {
mentionTarget: "#mentionElement",
userData: ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph']
};
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
#app {
color: #008cff;
height: 40px;
left: 15%;
position: absolute;
top: 10%;
width: 30%;
}
#comment {
font-size: 15px;
font-weight: 600;
}
#mentionElement {
min-height: 100px;
border: 1px solid #D7D7D7;
border-radius: 4px;
padding: 8px;
font-size: 14px;
width: 600px;
}
div#mentionElement[placeholder]:empty:before {
content: attr(placeholder);
color: #555;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <label id="comment" >Comments</label>
    <div id="mentionElement" placeholder = "Type @ and tag user"></div>
    <ejs-mention id='defaultMention' :target='mentionTarget'
:dataSource='userData'></ejs-mention>
  </div>
</template>
<script setup>
import { MentionComponent as EjsMention } from "@syncfusion/ej2-vue-
dropdowns";
const mentionTarget = "#mentionElement";
const userData = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
  #app {
      color: #008cff;
      height: 40px;
      left: 15%;
      position: absolute;
      top: 10%;
      width: 30%;
  }
  #comment {
      font-size: 15px;
      font-weight: 600;
  }
  #mentionElement {
      min-height: 100px;
      border: 1px solid #D7D7D7;
      border-radius: 4px;
      padding: 8px;
      font-size: 14px;
      width: 600px;
  }
  div#mentionElement[placeholder]:empty:before {
      content: attr(placeholder);
      color: #555;
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    <label id="comment" >Comments</label>
    <div id="mentionElement" placeholder = "Type @ and tag user"></div>
    <ejs-mention id='defaultMention' :target='mentionTarget'
:dataSource='userData'></ejs-mention>
  </div>
</template>
<script>
import { MentionComponent } from "@syncfusion/ej2-vue-dropdowns";
export default {
  components: {
    'ejs-mention': MentionComponent
  },
  name: 'app',
  data: function() {
    return {
      mentionTarget: "#mentionElement",
      userData: ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph']
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
  #app {
      color: #008cff;
      height: 40px;
      left: 15%;
      position: absolute;
      top: 10%;
      width: 30%;
  }
  #comment {
      font-size: 15px;
      font-weight: 600;
  }
  #mentionElement {
      min-height: 100px;
      border: 1px solid #D7D7D7;
      border-radius: 4px;
      padding: 8px;
      font-size: 14px;
      width: 600px;
  }
  div#mentionElement[placeholder]:empty:before {
      content: attr(placeholder);
      color: #555;
  }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/mention/getting-started-cs1" %}

Display Mention character

By using the showMentionChar property, the text content can be displayed along with the mention character. You can customize the mention character by using the mentionChar property in the Mention component.

By default, the mentionChar is @ and the showMentionChar property is disabled.

The following example displays the text content along with the mention character configured as #.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <label id="comment" >Comments</label>
    <div id="mentionElement" placeholder = "Type # and tag user"></div>
    <ejs-mention id='defaultMention' showMentionChar='true' mentionChar='#'
:target='mentionTarget' :dataSource='userData'></ejs-mention>
  </div>
</template>
<script setup>
import { MentionComponent as EjsMention } from "@syncfusion/ej2-vue-
dropdowns";
const mentionTarget = "#mentionElement";
const userData = ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'];
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
  #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  #comment {
      font-size: 15px;
      font-weight: 600;
  }
  #mentionElement {
      min-height: 100px;
      border: 1px solid #D7D7D7;
      border-radius: 4px;
      padding: 8px;
      font-size: 14px;
      width: 600px;
  }
  div#mentionElement[placeholder]:empty:before {
      content: attr(placeholder);
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <label id="comment" >Comments</label>
    <div id="mentionElement" placeholder = "Type # and tag user"></div>
    <ejs-mention id='defaultMention' showMentionChar='true' mentionChar='#'
:target='mentionTarget' :dataSource='userData'></ejs-mention>
  </div>
</template>
<script>
import { MentionComponent } from "@syncfusion/ej2-vue-dropdowns";
export default {
  components: {
    'ejs-mention': MentionComponent
  },
  name: 'app',
  data: function() {
    return {
      mentionTarget: "#mentionElement",
      userData: ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph']
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
  @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
  #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/getting-started-cs2" %}

Getting Started with the Vue Mention Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Mention component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

Copyright © 2001 -2024 Syncfusion Inc.

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Mention component as an example. To use the Vue Mention component in the project, the @syncfusion/ej2-vue-dropdowns package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-dropdowns --save

`

or

`bash

yarn add @syncfusion/ej2-vue-dropdowns

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Mention component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-list/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue Mention component using Composition API or Options API:

1.First, import and register the Mention component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

Copyright © 2001 -2024 Syncfusion Inc.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { MentionComponent as EjsMention } from "@syncfusion/ej2-vue-
dropdowns";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { MentionComponent } from "@syncfusion/ej2-vue-dropdowns";
//Component registeration
export default {
name: "App",
components: {
'ejs-mention' : MentionComponent,
}
}
</script>
```

2.In the template section, define the Mention component with the dataSource property and column definitions.

~/SRC/APP.VUE

```
<template>
<div class="control_wrapper">
<label style="font-size: 15px; font-weight: 600;">Comments</label>
<div id="mentionElement" style="min-height: 100px; border: 1px solid
#D7D7D7; border-radius: 4px; padding: 8px; font-size: 14px; width:
600px;"></div>
<ejs-mention id='defaultMention' :target='mentionTarget'
:dataSource='userData'></ejs-mention>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<label id="comment" >Comments</label>
<div id="mentionElement" placeholder = "Type # and tag user"></div>
<ejs-mention id='defaultMention' :target='data[0].mentionTarget'
:dataSource='data[0].userData'></ejs-mention>
</div>
</template>
<script setup>
import { MentionComponent as EjsMention } from "@syncfusion/ej2-vue-
dropdowns";
const data = [{ target: "#mentionElement",
userData: ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph'] }]
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
#app {
color: #008cff;
height: 40px;
left: 15%;
position: absolute;
top: 10%;
width: 30%;
}
#comment {
font-size: 15px;
font-weight: 600;
}
#mentionElement {
min-height: 100px;
border: 1px solid #D7D7D7;
border-radius: 4px;
padding: 8px;
font-size: 14px;
width: 600px;
}
div#mentionElement[placeholder]:empty:before {
content: attr(placeholder);
color: #555;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<label id="comment" >Comments</label>
<div id="mentionElement" placeholder = "Type # and tag user"></div>
<ejs-mention id='defaultMention' :target='mentionTarget'
:dataSource='userData'></ejs-mention>
</div>
</template>
<script>
import { MentionComponent } from "@syncfusion/ej2-vue-dropdowns";
//Component registration
export default {
name: 'App',
components: {
"ejs-mention": MentionComponent
},
data () {
return {
target: "#mentionElement",
userData: ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph']
}
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
#app {
color: #008cff;
height: 40px;
left: 15%;
position: absolute;
top: 10%;
width: 30%;
}
#comment {
font-size: 15px;
font-weight: 600;
}
#mentionElement {
min-height: 100px;
border: 1px solid #D7D7D7;
border-radius: 4px;
padding: 8px;
font-size: 14px;
width: 600px;
}
div#mentionElement[placeholder]:empty:before {
content: attr(placeholder);
color: #555;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

Display Mention character

By using the showMentionChar property, the text content can be displayed along with the mention character. You can customize the mention character by using the mentionChar property in the Mention component.

By default, the mentionChar is @ and the showMentionChar property is disabled.

The following example code displays the text content along with the mention character configured as #.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<label id="comment" >Comments</label>
<div id="mentionElement" placeholder = "Type # and tag user"></div>
<ejs-mention id='defaultMention' showMentionChar='true' mentionChar='#'
:target='data[0].mentionTarget' :dataSource='data[0].userData'></ejs-
mention>
</div>
</template>
<script setup>
import { MentionComponent as EjsMention } from "@syncfusion/ej2-vue-
dropdowns";
const data = [{ mentionTarget: "#mentionElement",
userData: ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph']}]
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
#app {
color: #008cff;
height: 40px;
left: 15%;
position: absolute;
top: 10%;
width: 30%;
}
#comment {
font-size: 15px;
font-weight: 600;
}
#mentionElement {
min-height: 100px;
border: 1px solid #D7D7D7;
border-radius: 4px;
padding: 8px;
font-size: 14px;
width: 600px;
}
div#mentionElement[placeholder]:empty:before {
content: attr(placeholder);
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<label id="comment" >Comments</label>
<div id="mentionElement" placeholder = "Type # and tag user"></div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<ejs-mention id='defaultMention' showMentionChar='true' mentionChar='#'
:target='mentionTarget' :dataSource='userData'></ejs-mention>
</div>
</template>
<script>
import { MentionComponent } from "@syncfusion/ej2-vue-dropdowns";
//Component registration
export default {
name: 'App',
components: {
"ejs-mention": MentionComponent
},
data () {
return {
mentionTarget: "#mentionElement",
userData: ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph']
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/bootstrap5.css";
@import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
#app {
color: #008cff;
height: 40px;
left: 15%;
position: absolute;
top: 10%;
width: 30%;
}
#comment {
font-size: 15px;
font-weight: 600;
}
#mentionElement {
min-height: 100px;
border: 1px solid #D7D7D7;
border-radius: 4px;
padding: 8px;
font-size: 14px;
width: 600px;
}
div#mentionElement[placeholder]:empty:before {
content: attr(placeholder);
}
</style>
```

Working with data in Vue Mention component

The Mention loads the data either from local data sources or remote data services using the dataSource property. It supports the data type of either `array` or `DataManager`.

Copyright © 2001 -2024 Syncfusion Inc.

The Mention also supports different kinds of data services such as OData V4 and Web API, and data formats such as XML, JSON, and JSONP with the help of DataManager adaptors.

| Fields | Type | Description |

|------|------|-------------|

| text | string | Specifies the display text of each list item. |

| value | number or string | Specifies the hidden data value mapped to each list item that should contain a unique value. |

| groupBy | string | Specifies the category under which the list item has to be grouped. |

| iconCss | string | Specifies the icon class of each list item. |

When binding complex data to the Mention, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in three ways as described in the following.

Array of simple data

The Mention has provided support to load an array of primitive data such as strings and numbers. Here, both the value and text fields act the same.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag user"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='userData'></ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MentionPlugin);
export default {
    data: function() {
        return {
            target: "#mentionElement",
            userData: ['Selma Rose', 'Garth', 'Robert', 'William', 'Joseph']
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/working-with-data-cs1" %}

Array of JSON data

The Mention can generate its list items through an array of complex data. Therefore, the appropriate columns should be mapped to the fields property.

In the following example, ID column and Game column from complex data have been mapped to the value field and text field, respectively.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag sport"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='userData' :fields='fields' ></ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MentionPlugin);
export default {
    data: function() {
        return {
            userData:  [
                { ID: 'game1', Game: 'Badminton' },
                { ID: 'game2', Game: 'Football' },
                { ID: 'game3', Game: 'Tennis' },
                { ID: 'game4', Game: 'Hockey' },
                { ID: 'game5', Game: 'Basketball' }
            ],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            fields: { text: 'Game', value: 'ID' },
            target: "#mentionElement"
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/working-with-data-cs2" %}

Array of Complex data

The Mention can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the fields property.

In the following example, Code.ID column and Country.Name column from complex data have been mapped to the value field and text field, respectively.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag
country"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='userData' :fields='fields' ></ejs-mention>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MentionPlugin);
export default {
    data: function() {
        return {
            userData: [
                { Country: { Name: 'Australia' }, Code: { ID: 'AU' }},
                { Country: { Name: 'Bermuda' }, Code: { ID: 'BM' }},
                { Country: { Name: 'Canada'}, Code:{ ID: 'CA'} },
                { Country: { Name: 'Cameroon'}, Code:{ ID: 'CM'} },
                { Country: { Name: 'Denmark'}, Code:{ ID: 'DK' }},
                { Country: { Name: 'France'}, Code: { ID:'FR'} }
            ],
            fields: { text: 'Country.Name', value: 'Code.ID' },
            target: "#mentionElement"
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
        color: #555;
    }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/mention/working-with-data-cs3" %}

Binding remote data

The Mention supports retrieval of data from remote data services with the help of DataManager component. The query property is used to fetch the data from the database and bind it to the Mention component.

OData v4 adaptor - Binding OData v4 service

The ODataV4 is an improved version of OData protocols, and the DataManager can also retrieve and consume OData v4 services. For more details on OData v4 services, refer to the odata documentation. To bind OData v4 service, use the ODataV4Adaptor.

The following sample displays the first 6 contacts from Customers table of the Northwind Data Service.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag user"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='dataSource' :query='query' :fields='fields'
popupWidth='250px'></ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
Vue.use(MentionPlugin);
export default {
    data: function() {
        return {
            dataSource: new DataManager({
                url:
'https://services.odata.org/V4/Northwind/Northwind.svc/',
                adaptor: new ODataV4Adaptor,
                crossDomain: true
            }),
            query: new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6),
            fields: { text: 'ContactName', value: 'CustomerID' },
            target:'#mentionElement'
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/working-with-data-cs4" %}

Web API adaptor

You can use WebApiAdaptor to bind mention with Web API created using OData endpoint.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag user"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='dataSource' :fields='fields' :query='query'
popupWidth='250px'></ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { Query, DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
Vue.use(MentionPlugin);
export default {
    data: function() {
        return {
            dataSource: new DataManager({
                url: 'https://ej2services.syncfusion.com/production/web-
services/api/Employees',
                adaptor: new WebApiAdaptor,
                crossDomain: true
            }),
            query: new Query().select(['FirstName', 'EmployeeID']).take(7),
            fields: { text: 'FirstName', value: 'EmployeeID' },
            target:'#mentionElement'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
        color: #555;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/working-with-data-cs5" %}

Filtering data in Vue Mention component

The Mention component has built-in support to filter data items. The filter operation starts as soon as you start typing characters in the mention element.

Limit the minimum filter character

You can control the minimum length of user input to initiate the search action using minLength property. This can be useful if you have a very large list of data. The default value is 0, where suggestion the list opened as soon as the user inputs the mention character.

The remote request does not fetch the search data until the search key contains three characters as shown in the following example.

APP.VUE

```
 <template>
    <div id="app">
        <label id="comment" >Comments</label>
```

```
        <div id="mentionElement" placeholder = "Type @ and tag user"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='DataManager' :query='query' :fields='fields'
minLength='3'></ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
Vue.use(MentionPlugin);
export default {
        data: function() {
        return {
            DataManager : new DataManager({
                url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
                adaptor: new ODataV4Adaptor,
                crossDomain: true
            }),
            query: new Query().select(['ContactName',
'CustomerID']).take(7),
            // map the appropriate column
            fields: { text: 'ContactName', value: 'CustomerID' },
            minLength: 3,
            target: "#mentionElement"
        }
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment{
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement{
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/filtering-data-cs1" %}

Change the filter type

While filtering, you can change the filter type to Contains, StartsWith, or EndsWith in the filterType property. The default filter operator is Contains.

- StartsWith - Filter the items that begin with the specified text value.
- Contains - Filter the items that contain the specified text value.
- EndsWith - Filter the items that end with the specified text value.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag user"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='DataManager' :query='query' :fields='fields'
minLength='3'></ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
Vue.use(MentionPlugin);
export default {
        data: function() {
        return {
            DataManager : new DataManager({
            url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
            adaptor: new ODataV4Adaptor,
            crossDomain: true
            }),
            query: new Query().select(['ContactName',
'CustomerID']).take(7),
            fields: { text: 'ContactName', value: 'CustomerID' },
            target: "#mentionElement"
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
```

```
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment{
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement{
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/filtering-data-cs2" %}

Allow spacing between search

While filtering, you can allow the space in the middle of the mention the data in the data source by using the allowSpaces property. If the data source does not match with the mentioned element data, the popup will be hidden on the space key press. The default value of allowSpaces is false.

By default, the allowSpaces property is disabled, and the space ends the mention component search.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag user"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='userData' :fields='fields' allowSpaces='true' ></ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MentionPlugin);
export default {
        data: function() {
        return {
            target: "#mentionElement",
            userData: [
                { Name : "Andrew Fuller", ID : "1" },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                { Name : "Anne Dodsworth" , ID : "2" },
                { Name : "Janet Leverling" , ID :  "3" },
                { Name : "Laura Callahan" , ID : "4" },
                { Name : "Margaret Peacock" , ID : "5" }
            ],
            fields: {text:'Name'}
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/filtering-data-cs3" %}

Customize the suggestion item count

While filtering, you can customize the number of list items to be displayed in the suggestion list by using the suggestionCount property.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag user"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='userData' :fields='fields' suggestionCount='8'></ejs-mention>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MentionPlugin);
export default {
        data: function() {
        return {
            userData:[
                { Name: "Selma Rose", EmailId: "selma@gmail.com" },
                { Name: "Maria", EmailId: "maria@gmail.com" },
                { Name: "Russo Kay", EmailId: "russo@gmail.com" },
                { Name: "Robert", EmailId: "robert@gmail.com" },
                { Name: "Camden Kate",EmailId: "camden@gmail.com" },
                { Name: "Garth", EmailId: "garth@gmail.com" },
                { Name: "Andrew James", EmailId: "james@gmail.com" },
                { Name: "Olivia", EmailId: "olivia@gmail.com" },
                { Name: "Sophia", EmailId: "sophia@gmail.com" },
                { Name: "Margaret", EmailId: "margaret@gmail.com" },
                { Name: "Ursula Ann", EmailId: "ursula@gmail.com" },
                { Name: "Laura Grace", EmailId: "laura@gmail.com" },
                { Name: "Albert", EmailId: "albert@gmail.com" },
                { Name: "William", EmailId: "william@gmail.com" }
            ],
            fields: { text: 'Name' },
            target: "#mentionElement"
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/filtering-data-cs4" %}

Sorting in Vue Mention component

You can display the suggestions list items in a specific order. It has possible types as Ascending, Descending, and None in the sortOrder property.

- None - The data source is not sorted.
- Ascending - The data source is sorted in ascending order.
- Descending - The data source is sorted in descending order.

In the following sample, the popup list data is rendered in Descending order.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag sport"></div>
        <ejs-mention id='defaultMention' :target='target' :fields='fields'
:dataSource='userData' sortOrder='Descending'></ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MentionPlugin);
export default {
    data: function() {
        return {
            target: "#mentionElement",
            userData : [
                { ID : "game1" ,Game : "Badminton"},
                { ID : "game2" ,Game : "Football" },
                { ID : "game3" ,Game : "Tennis"},
                { ID : "game4" ,Game : "Hockey"},
                { ID : "game5" ,Game : "Basketball"},
                { ID : "game6" ,Game : "Cricket"}
            ],
            fields : {text:'Game', value:'ID'}
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/sorting-cs1" %}

Template in Vue Mention component

The Mention has been provided with several options to customize each suggestion list item, display item, and data loading indication.

Item template

The content of each list item in Mention can be customized using itemTemplate property.

APP.VUE

```
    <template>
        <div id="app">
            <label id="comment" >Comments</label>
            <div id="mentionElement" placeholder = "Type @ and tag
user"></div>
            <ejs-mention id='defaultMention' :target='target'
:dataSource='dataSource' sortOrder='Descending' :fields='fields'
:itemTemplate="'iTemplate'" :query='query' popupWidth='250px'>
            <template v-slot:iTemplate="{data}">
                <span><span>{{data.FirstName}}</span><span class
='city'>{{data.City}}</span></span>
            </template>
            </ejs-mention>
        </div>
    </template>
    <script>
    import Vue from "vue";
    import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-
data';
    Vue.use(MentionPlugin);
    export default {
        data: function() {
            return {
                dataSource: new DataManager({
                url:
'https://services.odata.org/V4/Northwind/Northwind.svc/',
                adaptor: new ODataV4Adaptor,
                crossDomain: true
                }),
                query: new Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(6),
                fields: { text: 'FirstName', value: 'EmployeeID' },
                target: "#mentionElement"
            };
        }
    }
    </script>
    <style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
   #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    .city {
        right: 15px;
        position: absolute;
    }
    </style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/mention/template-cs1" %}

Display template

You can customize the mentioned value's display appearance using the displayTemplate property.

In the following sample, the selected value is displayed as a combined text of both FirstName and City in the mention element, which is separated by a hyphen.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag user"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='dataSource' :fields='fields' :itemTemplate="'iTemplate'"
:displayTemplate="'dTemplate'" :query='query' sortOrder='Descending'
popupWidth='250px'>
        <template v-slot:iTemplate="{data}">
        <span><span>{{data.FirstName}}</span><span class
='city'>{{data.City}}</span></span>
        </template>
        <template v-slot:dTemplate="{data}">
        <span>{{data.FirstName}} - {{data.City}}</span>
        </template>
        </ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
Vue.use(MentionPlugin);
export default {
    data: function() {
        return {
            dataSource: new DataManager({
            url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
            adaptor: new ODataV4Adaptor,
            crossDomain: true
            }),
            query: new Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(6),
            fields: { text: 'FirstName', value: 'EmployeeID' },
            target: "#mentionElement"
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    .city {
        right: 15px;
        position: absolute;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/template-cs2" %}

No records template

You can show the custom design of the popup list content when no data is found and no matches are found on search with the help of noRecordsTemplate property.

In the following sample, popup list content displays the notification of no data available.

APP.VUE

```
<template>
        <div id="app">
            <label id="comment" >Comments</label>
            <div id="mentionElement" placeholder = "Type @ and tag
user"></div>
            <ejs-mention id='defaultMention' :target='target' :dataSource=[]
:noRecordsTemplate='noRecordsTemplate' ></ejs-mention>
        </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MentionPlugin);
export default {
    data: function() {
        return {
            noRecordsTemplate:"<span class='norecord'> NO DATA
AVAILABLE</span>",
```

```
            target: "#mentionElement"
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
        color: #555;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/template-cs3" %}

Spinner template

Display the customized waiting spinner, when data fetching takes time to load in the suggestion list by using the spinnerTemplate property.

In the following sample, customized spinner is shown while fetching the data from the service.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag user"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='dataSource' sortOrder='Ascending' :fields='fields'
:query='query' :spinnerTemplate="'sTemplate'" popupWidth='250px' >
        <template v-slot:sTemplate>
```

```
            <div class="spinner_loader"></div>
        </template>
        </ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
Vue.use(MentionPlugin);
export default {
    data: function() {
        return {
            dataSource: new DataManager({
                url:
'https://services.odata.org/V4/Northwind/Northwind.svc/',
                adaptor: new ODataV4Adaptor,
                crossDomain: true
            }),
            query: new Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(26),
            fields: { text: 'FirstName', value: 'EmployeeID' },
            target: "#mentionElement"
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    .spinner_loader {
        border: 10px solid #f3f3f3;
        border-radius: 50%;
        border-top: 10px solid #3498db;
        width: 5px;
        height: 5px;
        position: absolute;
        top: 6px;
        left: 85px;
        -webkit-animation: spin 2s linear infinite; /* Safari */
        animation: spin 2s linear infinite;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/template-cs4" %}

Localization in Vue Mention component

The Localization library allows you to localize static text content of the noRecordsTemplate properties according to the culture currently assigned to the Mention.

| Locale key | en-US (default) |

|------|------|

| noRecordsTemplate | No records found |

Loading translations

To load the translation object to your application, use the load function of the **L10n** class.

In the following sample, French culture is set to the mention component and no data is loaded. Hence, the noRecordsTemplate property displays its text in French culture initially.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag user"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='customerData' :fields='fields' :query='query' locale='fr-
BE'></ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { L10n, setCulture} from '@syncfusion/ej2-base';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
Vue.use(MentionPlugin);
export default {
    data: function() {
        L10n.load({
            'fr-BE': {
                'dropdowns': {
                    'noRecordsTemplate': "Aucun enregistrement trouvé"
                }
            }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        });
        return {
            customerData: new DataManager({
                url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
                adaptor: new ODataV4Adaptor,
                crossDomain: true
            }),
            fields: { text: 'ContactName', value: 'CustomerID' },
            query: new Query().select(['ContactName',
'CustomerID']).take(0),
            target: "#mentionElement"
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/localization-cs1" %}

Customization in Vue Mention component

Show or hide mention character

You can show mention character as prefix of selected item in mention component using showMentionChar property. The default value of showMentionChar is false.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag user"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='userData' :fields='fields' showMentionChar='true'></ejs-
mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MentionPlugin);
export default {
    data: function() {
        return {
            target: "#mentionElement",
            userData: [
                { Name : "Selma Rose", EmailId : "selma@gmail.com"},
                { Name : "Maria", EmailId : "maria@gmail.com" },
                { Name : "Russo kay", EmailId : "russo@gmail.com" },
                { Name : "Robert", EmailId : "robert@gmail.com" },
                { Name : "Garth", EmailId : "garth@gmail.com" }
            ],
            fields: {text:'Name'}
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/customization-cs1" %}

Adding the suffix character after selection

The Mention has provided support to specify the custom suffix to append alongside the mentioned selected item while inserting. You can append space or new line character as suffixText.

The following example displays the white space next to the text selected since the suffixText is configured as .

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag
country"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='userData' :fields='fields' showMentionChar='true'
:suffixText='suffixText'></ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MentionPlugin);
export default {
        data: function() {
        return {
            target: "#mentionElement",
            suffixText: ' ',
            userData: [
                { Country : "Australia", Code : "AU" },
                { Country : "Bermuda" , Code : "BM" },
                { Country : "Canada" , Code :  "CA" },
                { Country : "Cameroon" , Code : "CM" },
                { Country : "Denmark" , Code : "DK" }
            ],
            fields: {text:'Country'}
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
```

```
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/customization-cs2" %}

Configure the popup list

You can customize the suggestion list width and height using the popupHeight and popupWidth properties.

By default, the popup list width value is set as auto. Depending on the mentioned suggestion data list, the width value is automatically adjusted. The popup list height value is set as 300px.

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag sport"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='userData' :fields='fields' popupHeight='200px'
popupWidth='250px'></ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MentionPlugin);
export default {
    data: function() {
        return {
            target: "#mentionElement",
            userData: [
                { ID : "game1" ,Game : "Badminton"},
                { ID : "game2" ,Game : "Football" },
                { ID : "game3" ,Game : "Tennis"},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                { ID : "game4" ,Game : "Hockey"},
                { ID : "game5" ,Game : "Basketball"},
                { ID : "game6" ,Game : "Cricket"}
            ],
            fields: {text:'Game'}
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/customization-cs3" %}

Trigger character

You can customize the trigger character by using the mentionChar property in the Mention component. The trigger character triggers the suggestion list to display in the target area.

By default, the mentionChar is @.

Accessibility in Vue Mention component

Web accessibility makes web content and web applications more accessible for people with disabilities. Mention component provides built-in compliance with WAI-ARIA specifications. The WAI-ARIA support is achieved using the attributes such as aria-selected and aria-activedescendent.

Copyright © 2001 -2024 Syncfusion Inc.

The Mention component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Mention component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Mention component uses the `Listbox` role where each list item has an `option` role. The following `ARIA attributes` denote the Mention state.

Copyright © 2001 -2024 Syncfusion Inc.

| **Properties** | **Functionalities** |

| --- | --- |

| aria-selected | Indicates the selected option.|

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

Keyboard interaction

You can use the following key shortcuts to access the Mention without interruptions.

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Arrow Down | Selects the first item in the Mention list. Otherwise, selects the item next to the currently selected item. |

| Arrow Up | Selects the item previous to the currently selected one. |

| Esc(Escape) | Closes the popup list when it is in an open state. |

| Enter | Selects the focused item, and when it is in an open state the popup list closes. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, inserts the selected popup list item and closes the popup list. |

APP.VUE

```
<template>
    <div id="app">
        <label id="comment" >Comments</label>
        <div id="mentionElement" placeholder = "Type @ and tag user"></div>
        <ejs-mention id='defaultMention' :target='target'
:dataSource='userData' :fields='fields'></ejs-mention>
    </div>
</template>
<script>
import Vue from "vue";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MentionPlugin);
export default {
    data: function() {
        return {
            target: "#mentionElement",
            userData: [
                { Name : "Andrew Fuller", ID : "1" },
                { Name : "Anne Dodsworth" , ID : "2" },
                { Name : "Janet Leverling" , ID :  "3" },
                { Name : "Laura Callahan" , ID : "4" },
                { Name : "Margaret Peacock" , ID : "5" }
            ],
          fields: {text:'Name'}
        };
    }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-list/styles/bootstrap5.css";
    @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 15%;
        position: absolute;
        top: 10%;
        width: 30%;
    }
    #comment {
        font-size: 15px;
        font-weight: 600;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/mention/Accessibility-cs1" %}

Ensuring accessibility

The Mention component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Mention component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Mention component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/mention.html" %}

See also

- Accessibility in Syncfusion Vue components

Menu

Getting Started with the Vue Menu Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Menu component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies are required to use the Menu component in your application.

```
|-- @syncfusion/ej2-vue-navigations
|-- @syncfusion/ej2-vue-base
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
```

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

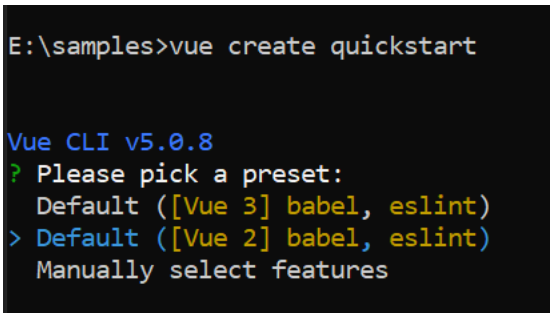
```bash
npm install -g @vue/cli
vue create quickstart
cd quickstart
npm run serve
```

or

```bash
yarn global add @vue/cli
vue create quickstart
cd quickstart
yarn run serve
```

When creating a new project, choose the option `Default ([Vue 2] babel, eslint)` from the menu.

Copyright © 2001 -2024 Syncfusion Inc.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Install Syncfusion Menu packages using below command.

`bash

npm install @syncfusion/ej2-vue-navigations --save

`

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Menu component as an example. Install the @syncfusion/ej2-vue-navigations package by running the following command:

`bash

npm install @syncfusion/ej2-vue-navigations --save

`

or

`bash

yarn add @syncfusion/ej2-vue-navigations

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Menu component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Add Syncfusion Vue component

Follow the below steps to add the Vue Menu component using Composition API or Options API:

1\. First, import and register the Menu component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { MenuComponent as EjsMenu } from "@syncfusion/ej2-vue-navigations";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { MenuComponent } from "@syncfusion/ej2-vue-navigations";
export default {
components: {
'ejs-menu': MenuComponent
}
}
</script>
```

Initialize Menu

Add the EJ2 Vue Menu using <ejs-menu> to the <template> section of the App.vue file in src directory.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div>
<ejs-menu :items='menuItems'></ejs-menu>
</div>
</template>
<script setup>
import { MenuComponent as EjsMenu } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu items definition
const menuItems =  [
{
text: 'File',
items: [
{ text: 'Open' },
{ text: 'Save' },
{ text: 'Exit' }
]
},
{
text: 'Edit',
items: [
{ text: 'Cut' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ text: 'Copy' },
{ text: 'Paste' }
]
},
{
text: 'View',
items: [
{ text: 'Toolbar' },
{ text: 'Sidebar' }
]
},
{
text: 'Tools',
items: [
{ text: 'Spelling & Grammar' },
{ text: 'Customize' },
{ text: 'Options' }
]
},
{ text: 'Go' },
{ text: 'Help' }
];
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div>
<ejs-menu :items='menuItems'></ejs-menu>
</div>
</template>
<script>
import { MenuComponent } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
components: {
'ejs-menu': MenuComponent
},
data: function() {
return {
//Menu items definition
menuItems:  [
{
text: 'File',
items: [
{ text: 'Open' },
{ text: 'Save' },
{ text: 'Exit' }
]
},
{
text: 'Edit',
items: [
{ text: 'Cut' },
{ text: 'Copy' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ text: 'Paste' }
]
},
{
text: 'View',
items: [
{ text: 'Toolbar' },
{ text: 'Sidebar' }
]
},
{
text: 'Tools',
items: [
{ text: 'Spelling & Grammar' },
{ text: 'Customize' },
{ text: 'Options' }
]
},
{ text: 'Go' },
{ text: 'Help' }
]
};
}
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div>
<ejs-menu :items='menuItems'></ejs-menu>
</div>
</template>
<script setup>
import { MenuComponent as EjsMenu } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Menu items definition
const menuItems = [
        {
        text: 'File',
        items: [
            { text: 'Open' },
            { text: 'Save' },
            { text: 'Exit' }
        ]
    },
    {
        text: 'Edit',
        items: [
            { text: 'Cut' },
            { text: 'Copy' },
            { text: 'Paste' }
        ]
```

```
    },
    {
        text: 'View',
        items: [
            { text: 'Toolbar' },
            { text: 'Sidebar' }
        ]
    },
    {
        text: 'Tools',
        items: [
            { text: 'Spelling & Grammar' },
            { text: 'Customize' },
            { text: 'Options' }
        ]
    },
    { text: 'Go' },
    { text: 'Help' }
    ];

</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div>
<ejs-menu :items='menuItems'></ejs-menu>
</div>
</template>
<script>
import { MenuComponent } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
   components: {
    'ejs-menu': MenuComponent
   },
   data: function() {
        return {
            //Menu items definition
           menuItems:  [
        {
        text: 'File',
        items: [
            { text: 'Open' },
            { text: 'Save' },
            { text: 'Exit' }
```

```
            ]
        },
        {
            text: 'Edit',
            items: [
                { text: 'Cut' },
                { text: 'Copy' },
                { text: 'Paste' }
            ]
        },
        {
            text: 'View',
            items: [
                { text: 'Toolbar' },
                { text: 'Sidebar' }
            ]
        },
        {
            text: 'Tools',
            items: [
                { text: 'Spelling & Grammar' },
                { text: 'Customize' },
                { text: 'Options' }
            ]
        },
        { text: 'Go' },
        { text: 'Help' }
        ]
        };
        }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs6" %}

Group menu items with separator

The separators are both horizontal and vertical lines used to separate the menu items. You cannot select the separators, but you can enable separators to group the menu items using the separator property. The Open and Save sub menu items are grouped using the separator property in the following sample.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div>
<ejs-menu :items='menuItems'></ejs-menu>
</div>
</template>
<script setup>
import { MenuComponent as EjsMenu } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
const menuItems =  [
        {
        text: 'File',
        items: [
            { text: 'Open' },
            { text: 'Save' },
            { separator: true },
            { text: 'Exit' }
        ]
    },
    {
        text: 'Edit',
        items: [
            { text: 'Cut' },
            { text: 'Copy' },
            { text: 'Paste' }
        ]
    },
    {
        text: 'View',
        items: [
            { text: 'Toolbar' },
            { text: 'Sidebar' },
            { text: 'Full Screen' }
        ]
    },
    {
        text: 'Tools',
         items: [
            { text: 'Spelling & Grammar' },
            { text: 'Customize' },
            { text: 'Options' }
        ]
    },
    { text: 'Go' },
    { text: 'Help' }
];
```

```
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div>
<ejs-menu :items='menuItems'></ejs-menu>
</div>
</template>
<script>
import { MenuComponent } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
   components: {
    'ejs-menu': MenuComponent
   },
   data: function() {
        return {
           menuItems:  [
        {
        text: 'File',
        items: [
            { text: 'Open' },
            { text: 'Save' },
            { separator: true },
            { text: 'Exit' }
        ]
    },
    {
        text: 'Edit',
        items: [
            { text: 'Cut' },
            { text: 'Copy' },
            { text: 'Paste' }
        ]
    },
    {
        text: 'View',
        items: [
            { text: 'Toolbar' },
            { text: 'Sidebar' },
            { text: 'Full Screen' }
        ]
    },
    {
        text: 'Tools',
```

```
          items: [
              { text: 'Spelling & Grammar' },
              { text: 'Customize' },
              { text: 'Options' }
          ]
      },
      { text: 'Go' },
      { text: 'Help' }
      ]
      };
      }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs7" %}

The separator property should not be given along with

the other fields in the MenuItem. You can also enable the separator to group **horizontal** menu items.

See Also

- Create menu using data source
- Customize menu items using template support
- Integrating with Toolbar component

Getting Started with the Vue Menu Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Menu component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

Copyright © 2001 -2024 Syncfusion Inc.

`bash
npm create vite@latest
`

or

`bash
yarn create vite
`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as `my-project` for this article.

`bash
? Project name: » my-project
`

2.Select `Vue` as the framework. It will create a Vue 3 project.

`bash
? Select a framework: » - Use arrow-keys. Return to submit.
Vanilla
Vue
React
Preact
Lit
Svelte
Others
`

3.Choose `JavaScript` as the framework variant to build this Vite project using JavaScript and Vue.

`bash
? Select a variant: » - Use arrow-keys. Return to submit.
JavaScript
TypeScript
Customize with create-vue ↗
Nuxt ↗
`

Copyright © 2001 -2024 Syncfusion Inc.

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Menu component as an example. To use the Vue Menu component in the project, the @syncfusion/ej2-vue-navigations package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-navigations --save

`

or

`bash

yarn add @syncfusion/ej2-vue-navigations

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Menu component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue Menu component using Composition API or Options API:

1.First, import and register the Menu component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { MenuComponent as EjsMenu } from "@syncfusion/ej2-vue-navigations";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { MenuComponent } from "@syncfusion/ej2-vue-navigations";
//Component registration
export default {
name: "App",
components: {
"ejs-menu": MenuComponent
}
}
</script>
```

2.In the template section, define the Menu component with the items property.

~/SRC/APP.VUE

```
<template>
<ejs-menu :items='menuItems'></ejs-menu>
</template>
```

3.Declare the values for the items property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const menuItems =  [
{
text: 'File',
items: [
{ text: 'Open' },
{ text: 'Save' },
{ text: 'Exit' }
]
},
{
text: 'Edit',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
items: [
{ text: 'Cut' },
{ text: 'Copy' },
{ text: 'Paste' }
]
},
{
text: 'View',
items: [
{ text: 'Toolbar' },
{ text: 'Sidebar' }
]
},
{
text: 'Tools',
items: [
{ text: 'Spelling & Grammar' },
{ text: 'Customize' },
{ text: 'Options' }
]
},
{ text: 'Go' },
{ text: 'Help' }
];
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
menuItems:  [
{
text: 'File',
items: [
{ text: 'Open' },
{ text: 'Save' },
{ text: 'Exit' }
]
},
{
text: 'Edit',
items: [
{ text: 'Cut' },
{ text: 'Copy' },
{ text: 'Paste' }
]
},
{
text: 'View',
items: [
{ text: 'Toolbar' },
{ text: 'Sidebar' }
]
},
{
```

```
text: 'Tools',
items: [
{ text: 'Spelling & Grammar' },
{ text: 'Customize' },
{ text: 'Options' }
]
},
{ text: 'Go' },
{ text: 'Help' }
],
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-menu :items='menuItems'></ejs-menu>
</template>
<script setup>
import { MenuComponent as EjsMenu } from "@syncfusion/ej2-vue-navigations";
const menuItems =  [
{
text: 'File',
items: [
{ text: 'Open' },
{ text: 'Save' },
{ text: 'Exit' }
]
},
{
text: 'Edit',
items: [
{ text: 'Cut' },
{ text: 'Copy' },
{ text: 'Paste' }
]
},
{
text: 'View',
items: [
{ text: 'Toolbar' },
{ text: 'Sidebar' }
]
},
{
text: 'Tools',
items: [
{ text: 'Spelling & Grammar' },
{ text: 'Customize' },
{ text: 'Options' }
]
},
{ text: 'Go' },
```

```
{ text: 'Help' }
];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-menu :items='menuItems'></ejs-menu>
</template>
<script>
import { MenuComponent } from "@syncfusion/ej2-vue-navigations";
// Component registration
export default {
name: "App",
// Declaring component and its directives
components: {
"ejs-menu": MenuComponent
},
// Bound properties declarations
data() {
return {
menuItems:  [
{
text: 'File',
items: [
{ text: 'Open' },
{ text: 'Save' },
{ text: 'Exit' }
]
},
{
text: 'Edit',
items: [
{ text: 'Cut' },
{ text: 'Copy' },
{ text: 'Paste' }
]
},
{
text: 'View',
items: [
{ text: 'Toolbar' },
{ text: 'Sidebar' }
]
},
{
text: 'Tools',
items: [
{ text: 'Spelling & Grammar' },
{ text: 'Customize' },
{ text: 'Options' }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
]
},
{ text: 'Go' },
{ text: 'Help' }
],
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

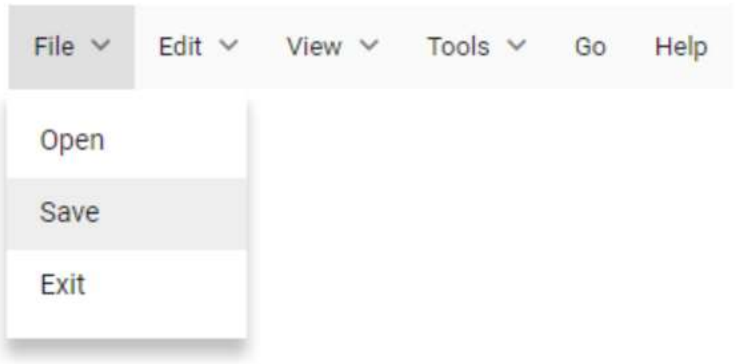
or

`bash

yarn run dev

`

The output will appear as follows:



For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Copyright © 2001 -2024 Syncfusion Inc.

Icons and sub menu items in Vue Menu component

Icons

The menu item contains an icon/image in it to provide a visual representation of an action. To place the icon on a menu item, set the iconCss property with the required icon CSS. By default, the icon is positioned at the left of the menu item. In the following sample, the icons of File and Edit menu items and Open, Save, Cut, Copy,and Paste sub menu items are added using the iconCss property.

APP.VUE

```
<template>
<div>
<ejs-menu :items='menuItems'></ejs-menu>
</div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
export default {
  data: function() {
    return {
        //Menu items definition
        menuItems:  [
            {
                text: 'File',
                iconCss: 'em-icons e-file',
                items: [
                    { text: 'Open', iconCss: 'em-icons e-open' },
                    { text: 'Save', iconCss: 'e-icons e-save' },
                    { separator: true },
                    { text: 'Exit' }
                ]
            },
            {
                text: 'Edit',
                iconCss: 'em-icons e-edit',
                items: [
                    { text: 'Cut', iconCss: 'em-icons e-cut' },
                    { text: 'Copy', iconCss: 'em-icons e-copy' },
                    { text: 'Paste', iconCss: 'em-icons e-paste' }
                ]
            },
            {
                text: 'View',
                items: [
                    { text: 'Toolbar' },
                    { text: 'Sidebar' },
                    { text: 'Full Screen' }
                ]
            },
            {
                text: 'Tools',
                items: [
                    { text: 'Spelling & Grammar' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    { text: 'Customize' },
                    { text: 'Options' }
                ]
            },
            { text: 'Go' },
            { text: 'Help' }
        ]
    };
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
@font-face {
    font-family: 'e-menu';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMjvJSpgAAAEoAAAAVmNtYXBsm2feAAABpAAAAGxnbHlmmEc
yrQAAAiQAAAWIaGVhZBJ0bwcAAADQAAAANmhoZWEHmQNyAAArAAAACRobXR4I0AAAAAAAYAAAAA
kbG9jYQaGB+4AAAIQAAAAFG1heHABGACaAAABCAAAACBuYW1lc0cOBgAAB6wAAAIlcG9zdJbKd4k
AAAnUAAAAfQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAACQABAAAAAQAAhka7o18
PPPUACwPoAAAAANe2FRwAAAAA17YVHAAAAAAD6gPqAAAACAACAAAAAAAAAAAEAAAAJAI4ABQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPrAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5anohgNS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABYAAAADgA
IAAIABuWp5bPluefo6CLohv//AADlqeWy5bnn6Ogi6IX//wAAAAAAAAAAAAAAAAAABAA4ADgAQABA
AEAAQAAAABQAGAAcACAABAAIAAwAEAAAAAACsASoBRAGwAhICUAKEAsQABAAAAAAD6gNZAD8AfwC
DAI0AAAEzHw0dAQ8OLw8/DiMzHw0dAQ8OLw8/DgMhAyEBIRU3EyUVBQMjAwgJCAgIBwcHBgUFBAQ
DAgEBAgMEBAUFBgcHBwgICAkJCAgIBwcHBgUFBAQDAgEBAQECAwQEBQUGBwcHCAgI5AgJCAgHBwc
GBQUEBAIDAQEDAgQEBQUGBwcHCAgJCAkICAgIBwYGBQUFAwMCAQEBAQIDAwUFBQYGBwgICAijAny
Q/qj+EgEKAssBcf5Yy9UBTwICAgQEBQYGBgcHCAgJCAkICQcIBwYGBgQFAwMCAQEBAQIDAwUEBgY
GBwgHCQgJCAkICAcHBgYGBQQEAgICAgICBAQFBgYGBwcICAkICQgJBwgHBgYGBAUDAwIBAQEBAgM
DBQQGBgYHCAcJCAkICQgIBwcGBgYFBAQCAgIBu/67AZUBAf5LAj0CAbUAAAAFAAAAAAPqA+oAAgA
WABgAPABkAAA3OQEnMx8PFQc3MQE7AR8OAQcvDwEzHwoPBi8PPwP/nAgODg4NDAwLCwoICAcFBAM
C6k4CdAgHEA4PDQ0MDAoJCAcGBAIB/kWFAQMEBgcJCgsLDQ0NDg4ODgLaBg0GBgYGBjwFBAMBAQE
CAgYJNAEDBAYHCQoKDAwNDQ4ODg40GQkKZJsBAwQFBwcJCQoMCw0NDg8OCE7pAnUDBQYHCQkLDAw
NDg0ODg7+SIgODg4NDg0MDAsKCAgGBAMBArUCAgMDBQU9CQkJCQgICAcNDjQNDg4ODQ0MDAsJCQc
GBAMBNA4DAgAAAAABAAAAAAPqA60ACgAAEyEVIRUhAxMhAyEVAcwBzPzEN5MDHrj84gOtXFz9/QG
n/boAAAAABQAAAAADjgPqAAMABwALAA8AUwAAEyEVITUhFSE1IRUhJxEhESUhHw8RDw8hLw8RPw7
qAij92AIo/dgCKP3YOwKi/XICeggICAgHBwYGBQUEAwMCAQEBAQIDAwQFBQYGBwcICAgI/YYICAg
IBwcGBgUFBAMDAgEBAQECAwMEBQUGBgcHCAgIAQs+9j72Prj9XgKi9gEBAgMDBAUFBgYHBwgICAj
8zggICAgHBwYGBQUEAwMCAQEBAQIDAwQFBQYGBwcICAgIAzIICAgIBwcGBgUFBAMDAgEABQAAAAA
DqQOpAAQACgAUAB4AOwAACQEXATUBFAcmNDIDBgcuATQ2MhYUAwYHLgE0NjIWFBc2NS4BIgYUFhc
yNxcHJiMOARQWMjY3NCc3ATM1Ayb++FkBMv5fFRUq3xglJjExSzEZGCUmMTFLMUoOAmKUY2JLJyF
mZiEnS2JilWICDmcBM4MDgP74WQE2K/50FQICKv6lGQICMkoyMkoB9xkCAjJKMjJKIyEnSmNjlGM
CDmdnDgJjlGNjSichZ/7NKwAAAAMAAAAA4oD5gAHABAAJwAAARUhNTMRIRElHgEGIiY0NjInBgc
jIgYVERQWMyEyNjURNCYrAS4BIgEZAbZd/ZABWAwBGiYZGSZhIg+8JjU1JgJ2JjU1JrwPRFgDLn5
9/TICz1IMJxkZJxlAGSkzJv0pJzMzJwLXJjMpMwADAAAAAAOpA+cAAwAUAB4AAAERIREnBhURFBY
XIT4BNRE0JiMhIicGFREzESE1IQYDTP4MQxs2JgH3JzU1J/4JJtgZXQIT/egmAs/9jwJxRBkm/Yc
mMwICMyYCeSYynxon/Y8CcV4CAAIAAAAAA+cD5wALACMAAAEOAQcuASc+ATceAQUeARcyNj8BARY
yNjQnATc+ATUuAScOAQLYA7SHiLMEBLOIh7T9KwXnrkeBNAMBAQ4kHA7+/wMpLgTora7nAk6HtAM
```

```
DtIeIswQEs4it6AQuKQP+/w4bJQ4BAQM0gUeu5wUF5wAAAAASAN4AAQAAAAAAAAABAAAAAQAAAAA
AAQAHAAEAAQAAAAAAAgAHAAgAAQAAAAAAAwAHAA8AAQAAAAAABAAHABYAAQAAAAAABQALAB0AAQA
AAAAABgAHACgAAQAAAAAACgAsAC8AAQAAAAAACwASAFsAAwABBAkAAAACAG0AAwABBAkAAQAOAG8
AAwABBAkAAgAOAH0AAwABBAkAAwAOAIsAAwABBAkABAAOAJkAAwABBAkABQAWAKcAAwABBAkABgA
OAL0AAwABBAkACgBYAMsAAwABBAkACwAkASMgZS1pY29uc1JlZ3VsYXJlLWljb25zZS1pY29uc1Z
lcnNpb24gMS4wZS1pY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R
1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGkAYwBvAG4AcwBSAGUAZwB1AGwAYQByAGUALQB
pAGMAbwBuAHMAZQAtAGkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAZQAtAGkAYwBvAG4
AcwBGAG8AbgB0ACAAZwBlAG4AZQByAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQB
zAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGk
AbwBuAC4AYwBvAG0AAAAAAgAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAJAQIBAwEEAQU
BBgEHAQgBCQEKAA1zaG9wcGluZy1jYXJ0B2VkaXQtMDUMZmlsZS1vcGVuLTAxDGZpbGUtdGV4dC0
wMQNDdXQFUGFzdGUEQ29weQZTZWFyY2gAAAAAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.em-icons {
    font-family: 'e-menu';
    font-style: normal;
    font-variant: normal;
    font-weight: normal;
    line-height: 1;
    text-transform: none;
}
.e-file::before {
    content: '\e886';
}
.e-edit::before {
    content: '\e822';
}
.e-cut::before {
    content: '\e5a9';
}
.e-copy::before {
    content: '\e5b3';
}
.e-paste::before {
    content: '\e5b2';
}
.e-open::before {
    content: '\e885';
}
.e-save::before {
    content: '\e98e';
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs18" %}

Navigation

Navigation in Menu is used to navigate to the other web page when a menu item is clicked. It can be achieved by providing a link to the menu item using the `url` property. In the following sample, the Navigation URL is added to sub menu items using the `url` property.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<div>
<ejs-menu :items='menuItems' :beforeItemRender='onBeforeItemRender'></ejs-
menu>
</div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin, MenuEventArgs } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
export default {
  data: function() {
    return {
        //Menu items definition
        menuItems:  [
            {
                text: 'Appliances',
                items: [
                    { text: 'Washing Machine', url:
'https://www.google.com/search?q=washing+machine' },
                    { text: 'Air Conditioners', url:
'https://www.google.com/search?q=air+conditioners' }
                ]
            },
            {
                text: 'Mobile',
                items: [
                    { text: 'Headphones', url:
'https://www.google.com/search?q=headphones' },
                    { text: 'Memory Cards', url:
'https://www.google.com/search?q=memory+cards' },
                    { text: 'Power Banks', url:
'https://www.google.com/search?q=power+banks' }
                ]
            },
            {
                text: 'Entertainment',
                items: [
                    { text: 'Televisions', url:
'https://www.google.com/search?q=televisions' },
                    { text: 'Home Theatres', url:
'https://www.google.com/search?q=home+theatres' },
                    { text: 'Gaming Laptops', url:
'https://www.google.com/search?q=gaming+laptops' }
                ]
            },
            { text: 'Fashion', url:
'https://www.google.com/search?q=fashion' },
            { text: 'Offers', url: 'https://www.google.com/search?q=offers'
}
        ]
    };
  },
  methods: {
    onBeforeItemRender: (args: MenuEventArgs) {
```

```
      if (args.item.url) {
        args.element.getElementsByTagName('a')[0].setAttribute('target',
'_blank');
      }
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs19" %}

Multilevel nesting

The Menu supports multiple level nesting, and it can be achieved by mapping the items property inside the parent menuItems. In the following sample, three-level nesting of menu has been provided.

APP.VUE

```
<template>
<div>
<ejs-menu :items='menuItems'></ejs-menu>
</div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
export default {
  data: function() {
    return {
        //Menu items definition
        menuItems:   [
            {
                text: 'Fashion',
                items: [
                    {
                        text: 'Men Fashion',
                        items: [
                            {
                                text: 'Personal Care',
                                items: [
                                    { text: 'Trimmers' },
                                    { text:  'Shavers' }
                                ]
                            },
                            {
```

```
                                text: 'Clothing',
                                items: [
                                    { text: 'Shirts' },
                                    { text: 'Jackets' },
                                    { text: 'Track Suits' }
                                ]
                            },
                            { text: 'Footwear' }
                        ]
                    },
                    {
                        text: 'Women Fashion',
                        items: [
                            {
                                text: 'Clothing',
                                items: [
                                    { text: 'Kurtas' },
                                    { text: 'Salwars' },
                                    { text: 'Sarees' }
                                ]
                            },
                            {
                                text: 'Jewellery',
                                items: [
                                    { text: 'Nosepins' },
                                    { text:  'Anklets' }
                                ]
                            }
                        ]
                    }
                ]
            },
            {
                text: 'Home & Living',
                items: [
                    {
                        text: 'Washing Machine',
                        items: [
                            { text: 'Fully Automatic' },
                            { text: 'Semi Automatic' }
                        ]
                    },
                    {
                        text: 'Air Conditioners',
                        items: [
                            { text: 'Inverter ACs' },
                            { text: 'Split ACs' }
                        ]
                    }
                ]
            },
            { text: 'Accessories' },
            { text: 'Sports' },
            { text: 'Gaming' }
        ]
    };
}
```

```
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs20" %}

The below table represents the MenuItem properties and it's description.

Property Name | Type | Description

|iconCss|string|Defines class/multiple classes separated by a space for the menu Item that is used to include an icon. Menu Item can include font icon and sprite image.

|id|string|Specifies the id for menu item.

|separator|boolean|Specifies separator between the menu items. Separator are either horizontal or vertical lines used to group menu items.

|items|MenuItemModel[]|Specifies the sub menu items that is the array of MenuItem model/

|text|string|Specifies text for menu item.

|url|string|Specifies url for menu item that creates the anchor link to navigate to the url provided.

Data source binding and custom menu items in Vue Menu component

Data binding

The Menu supports data source bindings such as array of JavaScript objects that can be structured as either hierarchical or self-referential data.

Hierarchical data

The Menu can be populated with hierarchical data source by assigning it to the items property, and the fields with corresponding keys can be mapped to the fields property.

JSON data

The Menu can generate its menu items through an array of complex data source by mapping fields from the fields property.

APP.VUE

```
<template>
<div>
<ejs-menu :items='datasource' :fields='menuFields'></ejs-menu>
</div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(MenuPlugin);
export default {
   data: function() {
        return {
            //Menu datasource
           datasource:  [
            {
                continent: 'Asia',
                countries: [
                    {
                        country: 'China',
                        languages: [
                            { language: 'Chinese' },
                            { language: 'Cantonese' }
                        ]
                    },
                    {
                        country: 'India',
                        languages: [
                            { language: 'English' },
                            { language: 'Hindi' },
                            { language: 'Tamil' }
                        ]
                    },
                    {
                        country: 'Japan',
                        languages: [
                            { language: 'Japanese' }
                        ]
                    }
                ]
            },
            {
                continent: 'Africa',
                countries: [
                    {
                        country: 'Nigeria',
                        languages: [
                            { language: 'English' },
                            { language: 'Hausa' }
                        ]
                    },
                    {
                        country: 'Egypt',
                        languages: [
                            { language: 'Arabic' }
                        ]
                    },
                    {
                        country: 'South Africa',
                        languages: [
                            { language: 'Tswana' },
                            { language: 'Swati' }
                        ]
                    }
                ]
            },
```

```
            {
                continent: 'North America',
                countries: [
                    {
                        country: 'Canada',
                        languages: [
                            { language: 'French' }
                        ]
                    },
                    {
                        country: 'Mexico',
                        languages: [
                            { language: 'Spanish' }
                        ]
                    },
                    {
                        country: 'USA',
                        languages: [
                            { language: 'English' }
                        ]
                    }
                ]
            },
            {
                continent: 'South America',
                countries: [
                    {
                        country: 'Brazil',
                        languages: [
                            { language: 'Portuguese' }
                        ]
                    },
                    {
                        country: 'Colombia',
                        languages: [
                            { language: 'Spanish' }
                        ]
                    },
                    {
                        country: 'Argentina',
                        languages: [
                            { language: 'Spanish' }
                        ]
                    }
                ]
            },
            {
                continent: 'Oceania',
                countries: [
                    {
                        country: 'Australia'
                    },
                    {
                        country: 'New Zealand'
                    },
                    {
                        country: 'Samoa'
```

```
                    },
                ]
            }
        ],
        menuFields: {
            text: ['continent', 'country', 'language'],
            children: ['countries', 'languages']
        }
    };
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs2" %}

Data Service

In application level, remote data binding can be achieved using DataManager.

To create Menu, assign items property with resultant data from callback function.

APP.VUE

```
<template>
    <div class="template-menu-control">
        <ejs-menu v-if="items" :items="items" :fields="menuFields"></ejs-
menu>
    </div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { ODataV4Adaptor, DataManager, Query } from "@syncfusion/ej2-data";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
var SERVICE_URI = 'https://services.odata.org/V4/Northwind/Northwind.svc/';
var result;
export default {
   data: function() {
    return {
        items: this.menuItems(),
        menuFields: {
            text: ['FirstName','ShipName'],
            children: ['Orders']
        }
      }
   },
   methods: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    menuItems() {
        new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor(),
crossDomain: true })
        .executeQuery(
        new Query().from('Employees').take(5).hierarchy(
            new Query()
            .foreignKey('EmployeeID')
            .from('Orders').take(13),
            function() {
                return [1,2,3,4,5]
            }
        ))
        .then((e) => {
            this.items =  e.result
        });
        return this.items;
     }
   }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
.template-menu-control {
    margin-top: 100px;
    text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs3" %}

Self-referential data

Menu can be populated from self-referential data structure that contains array of JSON objects with `parentId` mapping.

In the following example, the **id**, **pId**, and **text** columns from self-referential data have been mapped to the **itemId**, **parentId**, and **text** fields, respectively.

APP.VUE

```
<template>
<div>
<ejs-menu :items='datasource' :fields='menuFields'></ejs-menu>
</div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
export default {
   data: function() {
        return {
            //Menu datasource
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            datasource:  [
        { id: 'parent1', text: 'Events' },
    { id: 'parent2', text: 'Movies' },
    { id: 'parent3', text: 'Directory' },
    { id: 'parent4', text: 'Queries', pId: null },
    { id: 'parent5', text: 'Services', pId: null },
    { id: 'parent6', text: 'Conferences', pId: 'parent1' },
    { id: 'parent7', text: 'Music', pId: 'parent1' },
    { id: 'parent8', text: 'Workshops', pId: 'parent1' },
    { id: 'parent9', text: 'Now Showing', pId: 'parent2' },
    { id: 'parent10', text: 'Coming Soon', pId: 'parent2' },
    { id: 'parent10', text: 'Media Gallery', pId: 'parent3' },
    { id: 'parent11', text: 'Newsletters', pId: 'parent3' },
    { id: 'parent12', text: 'Our Policy', pId: 'parent4' },
    { id: 'parent13', text: 'Site Map', pId: 'parent4' },
    { id: 'parent14', text: 'Pop', pId: 'parent7' },
    { id: 'parent15', text: 'Folk', pId: 'parent7' },
    { id: 'parent16', text: 'Classical', pId: 'parent7' }
    ],
    //Menu fields definition
    menuFields: {
    itemId:'id',
    text: 'text',
    parentId: 'pId'
}
    };
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs4" %}

Custom menu items

The Menu can be customized using Essential JS2 Template engine to render the elements.

To customize menu items in your application, set your customized template string to the template property. In the following example, the menu has been rendered with customized menu items.

Install Syncfusion Layouts and Navigations packages using below command.

`bash

npm install @syncfusion/ej2-vue-layouts --save

npm install @syncfusion/ej2-vue-navigations --save

`

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div class="template-menu-control">
        <ejs-menu :items="data" :fields="menuFields"
:template="menuTemplate"></ejs-menu>
    </div>
</template>
<script>
import Vue from "vue";
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(MenuPlugin);
export default {
  data: function() {
    var menutemplateVue = Vue.component("demo", {
    template: '  <span v-if="data.category"> {{data.category}} </span>'+
    ' <div v-else-if="data.value" style="width: 100%;display: flex;justify-
content: space-between;">'+
     '<img v-if="data.url" class="e-avatar e-avatar-small" v-
bind:style="image" />' +
       '<span style="width:100%;">{{data.value}}</span></div>' +
    '<span class="e-menu-template" v-else-if="data.support">'+
        '<ul>'+
        '<li v-bind:key="index" v-for="(val,index) of data.support">'+
        ' {{val.value}} ' +
            '<span v-if="val.count" class="e-badge e-badge-
success">{{val.count}}</span> '+
        ' </li> '+
        '</ul>'+
    '</span>'+
    '<div tabindex="0" class="e-card" v-else>'+
        '<div class="e-card-header">'+
            '<div class="e-card-header-caption">'+
                '<div class="e-card-header-title">About Us</div>'+
            '</div>'+
        '</div>'+
        '<div class="e-card-content">{{data.about.value}} </div>'+
        '<div class="e-card-actions">'+
            '<button class="e-btn e-outline">Read More </button>'+
        '</div>'+
    '</div>',
    data() {
        return {
        data: {}
        }
    },
    computed: {
        image: function() {
        return 'background-image:
url(https://ej2.syncfusion.com/react/demos/src/menu/images/' + this.data.url
+ '.png);';
        }
    }
    });
    return {
        data: [
            {
```

```
                category: 'Products',
                options: [
                    { value: 'JavaScript', url: 'javascript' },
                    { value: 'Angular', url: 'angular' },
                    { value: 'ASP.NET Core', url: 'core' },
                    { value: 'ASP.NET MVC', url: 'mvc' }
                ]
            },
            {
                category: 'Services',
                options: [
                    {
                        support: [
                            { value: 'Application Development', count:
'1200+' },
                            { value: 'Maintenance & Support', count: '3700+'
},
                            { value: 'Quality Assurance' },
                            { value: 'Cloud Integration', count: '900+' }
                        ]
                    }
                ]
            },
            {
                category: 'About Us',
                options: [
                    {
                        about: {
                            value: "We are on a mission to provide world-
class best software solutions for web, mobile and desktop platforms. Around
900+ applications are desgined and delivered to our customers to make
digital & strengthen their businesses."
                        }
                    }
                ]
            },
            { category: 'Careers' },
            { category: 'Sign In' }
        ],
        menuFields: {
            text: ['category', 'value'],
            children: ['options']
        },
        menuTemplate: function () {
            return {
                template : menutemplateVue
            }
        }
    };
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
@import "../node_modules/@syncfusion/ej2-layouts/styles/material.css";
/* ej2 Menu template styles */
.template-menu-control {
    margin-top: 45px;
    text-align: center;
}
/* Common customization */
.e-bigger .template-menu-control .e-menu-wrapper.e-control ul.e-ul .e-menu-
item,
.template-menu-control .e-menu-wrapper.e-control ul.e-ul .e-menu-item,
.template-menu-control .e-menu-wrapper.e-control ul.e-ul .e-menu-item .e-
menu-template {
    display: flex;
    height: auto;
    padding: 0;
}
/* Avatar customization */
.e-bigger .template-menu-control .e-menu-wrapper ul .e-menu-item .e-avatar {
    background-size: auto;
    text-indent: 40px;
}
.template-menu-control .e-menu-wrapper ul .e-menu-item .e-avatar {
    background-color: inherit;
    background-position: 0;
    background-size: 25px;
    color: inherit;
    font-size: inherit;
    height: inherit;
    justify-content: left;
    margin: 0 10px;
    width: 100%;
    text-indent: 35px;
}
/* Badge customization */
.template-menu-control .e-menu-wrapper ul .e-menu-item ul li {
    padding: 0 10px;
}
.e-bigger .template-menu-control .e-menu-wrapper ul .e-menu-item ul li .e-
badge {
    margin: 18px 0px 0px 10px;
}
.template-menu-control .e-menu-wrapper ul .e-menu-item ul li .e-badge {
    float: right;
    margin: 13px 0px 0px 10px;
}
.template-menu-control .e-menu-wrapper ul .e-menu-item ul li:hover {
    background-color: #eee;
}
/* Card customization */
.e-bigger .template-menu-control .e-menu-wrapper ul.e-ul .e-menu-item .e-
card {
    width: 320px;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.template-menu-control .e-menu-wrapper ul.e-ul .e-menu-item .e-card {
    width: 290px;
    font-size: inherit;
    cursor: default;
    background-color: inherit;
    border-color: transparent;
}
.template-menu-control .e-menu-wrapper ul.e-ul .e-menu-item .e-card .e-card-
content {
    white-space: normal;
    color: inherit;
    padding-top: 0;
    text-align: justify;
    font-size: inherit;
}
.e-bigger .template-menu-control .e-menu-wrapper ul.e-ul .e-menu-item .e-
card .cb-icons {
    width: 40px;
    font-size: 40px;
    height: 40px;
}
.template-menu-control .e-menu-wrapper ul.e-ul .e-menu-item .e-card .cb-
icons {
    width: 30px;
    font-size: 30px;
    height: 30px;
}
.template-menu-control .e-menu-wrapper ul.e-ul .e-menu-item .e-card .e-card-
btn {
    background-color: inherit;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs5" %}

See Also

- Render menu with items

Use case scenarios in Vue Menu component

Scrollable menu

The menu component supports horizontal and vertical scrolling to render large menus and submenus in an adaptive way. This can be achieved by enabling the enableScrolling property and by restricting the corresponding menu/submenu size.

APP.VUE

```
<template>
    <div>
        <ejs-menu :items='menuItems' :beforeOpen='onBeforeOpen' cssClass='e-
scrollable-menu' :enableScrolling='true'></ejs-menu>
    </div>
</template>
<script>
```

```
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple, closest } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
export default {
  data: function() {
    return {
        // Menu items definition
        menuItems: [
        {
            text: 'Appliances',
            items: [
                {
                    text: 'Kitchen',
                    items: [
                        { text: 'Electric Cookers' },
                        { text: 'Coffee Makers' },
                        { text: 'Blenders' },
                        { text: 'Microwave Ovens' }
                    ]
                },
                {
                    text: 'Television',
                    items: [
                        { text: 'Our Exclusive TVs' },
                        { text: 'Smart TVs' },
                        { text: 'Big Screen TVs' }
                    ]
                },
                {
                    text: 'Washing Machine'
                },
                {
                    text: 'Refrigerators'
                },
                {
                    text: 'Air Conditioners',
                    items: [
                        { text: 'Inverter ACs' },
                        { text: 'Split ACs' },
                        { text: 'Window ACs' },
                    ]
                },
                {
                    text: 'Water Purifiers'
                },
                {
                    text: 'Air Purifiers'
                },
                {
                    text: 'Chimneys'
                },
                {
                    text: 'Inverters'
                },
                {
```

```
                    text: 'Healthy Living'
                },
                {
                    text: 'Vacuum Cleaners'
                },
                {
                    text: 'Room Heaters'
                },
                {
                    text: 'New Launches'
                }
            ]
        },
        {
            text: 'Accessories',
            items: [
                {
                    text: 'Mobile',
                    items: [
                        { text: 'Headphones' },
                        { text: 'Memory Cards' },
                        { text: 'Power Banks' },
                        { text: 'Mobile Cases' },
                        { text: 'Screen Protectors' }
                    ]
                },
                {
                    text: 'Laptops'
                },
                {
                    text: 'Desktop PC',
                    items: [
                        { text: 'Pendrives' },
                        { text: 'External Hard Disks' },
                        { text: 'Monitors' },
                        { text: 'Keyboards' }
                    ]
                },
                {
                    text: 'Camera',
                    items: [
                        { text: 'Lens' },
                        { text: 'Tripods' }
                    ]
                }
            ]
        },
        {
            text: 'Fashion',
            items: [
                {
                    text: 'Men'
                },
                {
                    text: 'Women'
                }
            ]
```

```
        },
        {
            text: 'Home & Living',
            items: [
                {
                    text: 'Furniture'
                },
                {
                    text: 'Decor'
                },
                {
                    text: 'Smart Home Automation'
                },
                {
                    text: 'Dining & Serving'
                }
            ]
        },
        {
            text: 'Entertainment',
            items: [
                {
                    text: 'Televisions'
                },
                {
                    text: 'Home Theatres'
                },
                {
                    text: 'Gaming Laptops'
                }
            ]
        },
        {
            text: 'Contact Us'
        },
        {
            text: 'Help'
        }]
    };
  },
   methods: {
        onBeforeOpen(args) {
            // Restricting sub menu wrapper height
            if (args.parentItem.text === 'Appliances') {
                closest(args.element, '.e-menu-wrapper').style.height =
'230px';
            }
        }
   }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    text-align: center;
}
/* Restricting the parent menu wrapper size */
.e-menu-wrapper.e-scrollable-menu:not(.e-menu-popup) {
    width: 492px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs21" %}

Menu in toolbar

The following example demonstrates how to integrate Menu with Toolbar component.

APP.VUE

```
<template>
<div class="toolbar-menu-control">
<ejs-toolbar ref="toolbar" :created="onCreated">
    <e-items>
        <e-item :template='menuTemplate'></e-item>
        <e-item prefixIcon='em-icons e-shopping-cart' align='Right' ></e-
item>
    </e-items>
</ejs-toolbar>
</div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin, ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
Vue.use(ToolbarPlugin);
var menutemplateVue = Vue.component("demo1", {
  template: '<ejs-menu :items="menuItems"
:animationSettings="animation"></ejs-menu>',
  data() {
    return {
      data: {},
      menuItems: [
            {
                text: 'Appliances',
                items: [
                    {
                        text: 'Kitchen',
                        items: [
                            { text: 'Electric Cookers' },
                            { text: 'Coffee Makers' },
                            { text: 'Blenders' }
                        ]
                    },
                    {
                        text: 'Washing Machine',
                        items: [
                            { text: 'Fully Automatic' },
                            { text: 'Semi Automatic' }
```

```
                        ]
                    },
                    {
                        text: 'Air Conditioners',
                        items: [
                            { text: 'Inverter ACs' },
                            { text: 'Split ACs' },
                            { text: 'Window ACs' }
                        ]
                    }
                ]
            },
            {
                text: 'Accessories',
                items: [
                    {
                        text: 'Mobile',
                        items: [
                            { text: 'Headphones' },
                            { text: 'Memory Cards' },
                            { text: 'Power Banks' }
                        ]
                    },
                    {
                        text: 'Computer',
                        items: [
                            { text: 'Pendrives' },
                            { text: 'External Hard Disks' },
                            { text: 'Monitors' }
                        ]
                    }
                ]
            },
            {
                text: 'Fashion',
                items: [
                    {
                        text: 'Men',
                        items: [
                            { text: 'Shirts' },
                            { text: 'Jackets' },
                            { text: 'Track Suits' }
                        ]
                    },
                    {
                        text: 'Women',
                        items: [
                            { text: 'Kurtas' },
                            { text: 'Salwars' },
                            { text: 'Sarees' }
                        ]
                    }
                ]
            },
            {
                text: 'Home & Living',
                items: [
```

```
                        {
                            text: 'Furniture',
                            items: [
                                { text: 'Beds' },
                                { text: 'Mattresses' },
                                { text: 'Dining Tables' }
                            ]
                        },
                        {
                            text: 'Decor',
                            items: [
                                { text: 'Clocks' },
                                { text: 'Wall Decals' },
                                { text: 'Paintings' }
                            ]
                        }
                    ]
                }
            ],
          animation: { effect: 'none' }
        };
    }
});
export default {
   data: function() {
    return {
        menuTemplate: function() {
            return {
                template: menutemplateVue
            }
        }
    }
  },
  methods: {
      onCreated: function() {
          this.$refs.toolbar.$el.ej2_instances[0].refreshOverflow();
        }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
@font-face {
    font-family: 'e-menu';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMjvJSpgAAAEoAAAAVmNtYXBsm2feAAABpAAAAGxnbHlmmEc
yrQAAAiQAAAWIaGVhZBJ0bwcAAADQAAAANmhoZWEHmQNyAAArAAAACRobXR4I0AAAAAAAYAAAAA
kbG9jYQaGB+4AAAIQAAAAFG1heHABGACaAAABCAAAACBuYW1lc0cOBgAAB6wAAAIlcG9zdJbKd4k
AAAnUAAAAfQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAACQABAAAAAQAAhka7o18
PPPUACwPoAAAAANe2FRwAAAAA17YVHAAAAAAD6gPqAAAACAACAAAAAAAAAAEAAAAJAI4ABQAAAAA
```

```
AAgAAAAoACgAAAP8AAAAAAAAAAQPrAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5anohgNS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABYAAAADgA
IAAIABuWp5bPluefo6CLohv//AADlqeWy5bnn6Ogi6IX//wAAAAAAAAAAAAAAAAAABAA4ADgAQABA
AEAAQAAAABQAGAAcACAABAAIAAwAEAAAAAACsASoBRAGwAhICUAKEAsQABAAAAAAD6gNZAD8AfwC
DAI0AAAEzHw0dAQ8OLw8/DiMzHw0dAQ8OLw8/DgMhAyEBIRU3EyUVBQMjAwgJCAgIBwcHBgUFBAQ
DAgEBAgMEBAUFBgcHBwgICAkJCAgIBwcHBgUFBAQDAgEBAQECAwQEBQUGBwcHCAgI5AgJCAgHBwc
GBQUEBAIDAQEDAgQEBQUGBwcHCAgJCAkICAgIBwYGBQUFAwMCAQEBAQIDAwUFBQYGBwgICAijAny
Q/qj+EgEKAssBcf5Yy9UBTwICAgQEBQYGBgcHCAgJCAkICQcIBwYGBgQFAwMCAQEBAQIDAwUEBgY
GBwgHCQgJCAkICAcHBgYGBQQEAgICAgICBAQFBgYGBwcICAkICQgJBwgHBgYGBAUDAwIBAQEBAgM
DBQQGBgYHCAcJCAkICQgIBwcGBgYFBAQCAgIBu/67AZUBAf5LAj0CAbUAAAAFAAAAAAPqA+oAAgA
WABgAPABkAAA3OQEnMx8PFQc3MQE7AR8OAQcvDwEzHwoPBi8PPwP/nAgODg4NDAwLCwoICAcFBAM
C6k4CdAgHEA4PDQ0MDAoJCAcGBAIB/kWFAQMEBgcJCgsLDQ0NDg4ODgLaBg0GBgYGBjwFBAMBAQE
CAgYJNAEDBAYHCQoKDAwNDQ4ODg40GQkKZJsBAwQFBwcJCQoMCw0NDg8OCE7pAnUDBQYHCQkLDAw
NDg0ODg7+SIgODg4NDg0MDAsKCAgGBAMBArUCAgMDBQU9CQkJCQgICAcNDjQNDg4ODQ0MDAsJCQc
GBAMBNA4DAgAAAAABAAAAAAPqA60ACgAAEyEVIRUhAxMhAyEVAcwBzPzEN5MDHrj84gOtXFz9/QG
n/boAAAAABQAAAAADjgPqAAMABwALAA8AUwAAEyEVITUhFSE1IRUhJxEhESUhHw8RDw8hLw8RPw7
qAij92AIo/dgCKP3YOwKi/XICeggICAgHBwYGBQUEAwMCAQEBAQIDAwQFBQYGBwcICAgI/YYICAg
IBwcGBgUFBAMDAgEBAQECAwMEBQUGBgcHCAgIAQs+9j72Prj9XgKi9gEBAgMDBAUFBgYHBwgICAj
8zggICAgHBwYGBQUEAwMCAQEBAQIDAwQFBQYGBwcICAgIAzIICAgIBwcGBgUFBAMDAgEABQAAAAA
DqQOpAAQACgAUAB4AOwAACQEXATUBFAcmNDIDBgcuATQ2MhYUAwYHLgE0NjIWFBc2NS4BIgYUFhc
yNxcHJiMOARQWMjY3NCc3ATM1Ayb++FkBMv5fFRUq3xglJjExSzEZGCUmMTFLMUoOAmKUY2JLJyF
mZiEnS2JilWICDmcBM4MDgP74WQE2K/50FQICKv6lGQICMkoyMkoB9xkCAjJKMjJKIyEnSmNjlGM
CDmdnDgJjlGNjSichZ/7NKwAAAAMAAAAAA4oD5gAHABAAJwAAARUhNTMRIRElHgEGIiY0NjInBgc
jIgYVERQWMyEyNjURNCYrAS4BIgEZAbZd/ZABWAwBGiYZGSZhIg+8JjU1JgJ2JjU1JrwPRFgDLn5
9/TICz1IMJxkZJxlAGSkzJv0pJzMzJwLXJjMpMwADAAAAAAOpA+cAAwAUAB4AAAERIREnBhURFBY
XIT4BNRE0JiMhIicGFREzESE1IQYDTP4MQxs2JgH3JzU1J/4JJtgZXQIT/egmAs/9jwJxRBkm/Yc
mMwICMyYCeSYynxon/Y8CcV4CAAIAAAAAA+cD5wALACMAAAEOAQcuASc+ATceAQUeARcyNj8BARY
yNjQnATc+ATUuAScOAQLYA7SHiLMEBLOIh7T9KwXnrkeBNAMBAQ4kHA7+/wMpLgTora7nAk6HtAM
DtIeIswQEs4it6AQuKQP+/w4bJQ4BAQM0gUeu5wUF5wAAAAASAN4AAQAAAAAAAAABAAAAAQAAAAA
AAQAHAAEAAQAAAAAAgAHAAgAAQAAAAAAwAHAA8AAQAAAAAABAAHABYAAQAAAAAABQALAB0AAQA
AAAAABgAHACgAAQAAAAACgAsAC8AAQAAAAACwASAFsAAwABBAkAAAACAG0AAwABBAkAAQAOAG8
AAwABBAkAAgAOAH0AAwABBAkAAwAOAIsAAwABBAkABAAOAJkAAwABBAkABQAWAKcAAwABBAkABgA
OAL0AAwABBAkACgBYAMsAAwABBAkACwAkASMgZS1pY29uc1JlZ3VsYXJlLWljb25zZS1pY29uc1Z
lcnNpb24gMS4wZS1pY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R
1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGkAYwBvAG4AcwBSAGUAZwB1AGwAYQByAGUALQB
pAGMAbwBuAHMAZQAtAGkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAZQAtAGkAYwBvAG4
AcwBGAG8AbgB0ACAAZwBlAG4AZQByAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQB
zAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGk
AbwBuAC4AYwBvAG0AAAAAAgAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAJAQIBAwEEAQU
BBgEHAQgBCQEKAA1zaG9wcGluZy1jYXJ0B2VkaXQtMDUMZmlsZS1vcGVuLTAxDGZpbGUtdGV4dC0
wMQNDdXQFUGFzdGUEQ29weQZTZWFyY2gAAAAAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.em-icons {
    font-family: 'e-menu';
    font-style: normal;
    font-variant: normal;
    font-weight: normal;
    text-transform: none;
    line-height: 2;
}
.toolbar-menu-control .e-toolbar-items .e-toolbar-item .e-tbar-btn .e-btn-
icon.e-shopping-cart {
    font-size: 20px;
    margin-right: 1px;
```

```
}
.toolbar-menu-control .e-shopping-cart::before {
    content: '\e7e8';
}
.toolbar-menu-control .e-menu-wrapper .e-menu {
    border: none;
}
.toolbar-menu-control {
    margin: 45px auto 0;
    max-width: 950px;
}
.toolbar-menu-control .e-toolbar .e-toolbar-left .e-toolbar-item.e-template
{
    padding: 0;
}
.toolbar-menu-control .e-toolbar {
    overflow: visible !important;
}
.toolbar-menu-control .e-menu-wrapper {
    margin-right: 160px;
}
.toolbar-menu-control .e-hscroll .e-hscroll-content {
    position: static;
}
@media only screen and (max-width: 1300px) {
    .toolbar-menu-control {
        width: auto;
    }
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs22" %}

Hamburger menu

The following example demonstrates the use case of menu with Accordion component integrated in SideBar.

APP.VUE

```
<template>
<div>
    <div class="header">
        <span id="hamburger" class="e-icons menu default"
@click='hamburgerClick'></span>
        <div class="content">Header content</div>
    </div>
    <ejs-sidebar ref="sidebar" id='default-sidebar' width='220px'
type='Over'>
        <div class="title-header">
            <div style="display:inline-block"> Menu </div>
            <span  id="close" class="e-icons" @click='close'></span>
        </div>
        <div class="content-area">
            <ejs-accordion ref="accordion" :items='items'
:expanding='expand' :clicked='clicked'></ejs-accordion>
        </div>
```

```
    </ejs-sidebar>
</div>
</template>
<script>
import Vue from 'vue';
import { Accordion } from "@syncfusion/ej2-navigations";
import { SidebarPlugin, AccordionPlugin } from "@syncfusion/ej2-vue-
navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SidebarPlugin);
Vue.use(AccordionPlugin);
export default {
   data: function() {
    return {
                  items: [
        {
            header: 'Appliances',
            content: '<div id="Appliances_Items"></div>',
            subItems: [
                {
                    header: 'Kitchen',
                    content: '<div id="Appliances_Kitchen_Items"></div>',
                    subItems: [
                        { header: 'Electric Cookers' },
                        { header: 'Coffee Makers' },
                        { header: 'Blenders' },
                    ]
                },
                {
                    header: 'Washing Machine',
                    content: '<div id="Appliances_Washing_Items"></div>',
                    subItems: [
                        { header: 'Fully Automatic' },
                        { header: 'Semi Automatic' }
                    ]
                },
                {
                    header: 'Air Conditioners',
                    content: '<div
id="Appliances_Conditioners_Items"></div>',
                    subItems: [
                        { header: 'Inverter ACs' },
                        { header: 'Split ACs' },
                        { header: 'Window ACs' },
                    ]
                }
            ]
        },
        {
            header: 'Accessories',
            content: '<div id="Accessories_Items"></div>',
            subItems: [
                {
                    header: 'Mobile',
                    content: '<div id="Accessories_Mobile_Items"></div>',
                    subItems: [
```

```
                        { header: 'Headphones' },
                        { header: 'Memory Cards' },
                        { header: 'Power Banks' }
                    ]
                },
                {
                    header: 'Computer',
                    content: '<div id="Accessories_Computer_Items"></div>',
                    subItems: [
                        { header: 'Pendrives' },
                        { header: 'External Hard Disks' },
                        { header: 'Monitors' }
                    ]
                }
            ]
        },
        {
            header: 'Fashion',
            content: '<div id="Fashion_Items"></div>',
            subItems: [
                { header: 'Men' },
                { header: 'Women' }
            ]
        },
        {
            header: 'Home & Living',
            content: '<div id="Home_Living_Items"></div>',
            subItems: [
                { header: 'Furniture' },
                { header: 'Decor' }
            ]
        },
        {
            header: 'Entertainment',
            content: '<div id="Entertainment_Items"></div>',
            subItems: [
                { header: 'Televisions' },
                { header: 'Home Theatres' },
                { header: 'Gaming Laptops' }
            ]
        }
    ],
    };
  },
  methods: {
    expand: function(e) {
        if (e.isExpanded) {
            if (e.element.getElementsByClassName('e-acrdn-
content')[0].children[0].classList.contains('e-accordion')) {
                return;
            }
            //Initialize Nested Accordion component
            var nestAcrdn = new Accordion({
                items: e.item.subItems,
                expanding: this.expand,
                clicked: this.clicked
            });
```

```
            var elemId = e.element.getElementsByClassName('e-acrdn-
content')[0].children[0].id;
            //Render initialized Nested Accordion component
            nestAcrdn.appendTo('#' + elemId);
        }
    },
    clicked: function(e){
        if (!e.item && !(e.originalEvent.target.closest('.e-acrdn-
item').getElementsByClassName('e-tgl-collapse-icon').length)) {
            this.$refs.sidebar.hide();
        }
    },
    hamburgerClick: function() {
        this.$refs.sidebar.show();
    },
    close: function() {
        this.$refs.sidebar.hide();
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin: 0;
}
.header {
    width: 100%;
    background-color: #7b8cfb;
}
#default-sidebar,
.header .content {
    background-color: #7b8cfb;
    color: white;
    border: none;
}
.header .content {
    font-size: 20px;
    line-height: 50px;
    text-align: center;
}
.main-content {
    text-align: center;
    font-size: 20px;
    padding: 100px 15px;
}
#default-sidebar .close-btn:hover {
    color: rgba(0, 0, 0, .87);
    background-color: #fafafa;
}
#hamburger.menu {
    font-size: 25px;
    cursor: pointer;
    float: left;
    line-height: 50px;
    position: absolute;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    z-index: 1000;
    left: 25px;
    color: white;
}
#hamburger.menu:before {
    content: '\e99a';
}
#close:before {
    content: "\e945";
}
.title-header {
    text-align: center;
    font-size: 18px;
    padding: 15px 15px 35px;
}
.e-sidebar .title-header #close {
    cursor: pointer;
    line-height: 25px;
    float: right;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs23" %}

Mobile view

The following example demonstrates the use case of Menu in Mobile mode by using ListView component with hamburger.

Install Syncfusion Lists packages using below command.

`bash

npm install @syncfusion/ej2-vue-lists --save

`

APP.VUE

```
<template>
<div class="layoutWrapper">
    <div class="speaker">
        <div class="camera"></div>
    </div>
    <div class="layout">
        <div id="container">
            <div id="header">
                <span id="hamburger" @click="hamburgerClick" class="e-icons
menu default"></span>
                <div class="content">Header</div>
            </div>
            <!-- ListView element -->
            <ejs-listview  ref="listview" :dataSource="dataSource"
headerTitle="Menu" :showHeader="true" :select="onSelect" tabindex="1" v-
bind:style="{display: 'none'}" ></ejs-listview>
            <span id="close" @click="onClick" class="e-icons" v-
bind:style="{display: 'none'}"></span>
        </div>
```

```
    </div>
    <div class="outerButton"> </div>
</div>
</template>
<script>
import Vue from 'vue';
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import { enableRipple, Animation } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(ListViewPlugin);
export default {
   data: function() {
    return {
                dataSource: [
    {
      text: 'Appliances',
      id: 'list1',
      child: [
        {
          text: 'Kitchen',
          id: 'list1_1',
          child: [
            { id: 'list1_1_1', text: 'Electric Cookers' },
            { id: 'list1_1_2', text: 'Coffee Makers' },
            { id: 'list1_1_3', text: 'Blenders' },
          ]
        },
        {
          text: 'Washing Machine',
          id: 'list1_2',
          child: [
            { id: 'list1_2_1', text: 'Fully Automatic' },
            { id: 'list1_2_2', text: 'Semi Automatic' }
          ]
        },
        {
          text: 'Air Conditioners',
          id: 'list1_3',
          child: [
            { id: 'list1_3_1', text: 'Inverter ACs' },
            { id: 'list1_3_2', text: 'Split ACs' },
            { id: 'list1_3_3', text: 'Window ACs' },
          ]
        }
      ]
    },
    {
      text: 'Accessories',
      id: 'list2',
      child: [
        {
          text: 'Mobile',
          id: 'list2_1',
          child: [
            { id: 'list2_1_1', text: 'Headphones' },
            { id: 'list2_1_2', text: 'Memory Cards' },
            { id: 'list2_1_3', text: 'Power Banks' }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            ]
          },
          {
            text: 'Computer',
            id: 'list2_2',
            child: [
              { id: 'list2_2_1', text: 'Pendrives' },
              { id: 'list2_2_2', text: 'External Hard Disks' },
              { id: 'list2_2_3', text: 'Monitors' }
            ]
          }
        ]
      },
      {
        text: 'Fashion',
        id: 'list3',
        child: [
          { id: 'list3_1', text: 'Men' },
          { id: 'list3_2', text: 'Women' }
        ]
      },
      {
        text: 'Home & Living',
        id: 'list4',
        child: [
          { id: 'list4_1', text: 'Furniture' },
          { id: 'list4_2', text: 'Decor' }
        ]
      },
      {
        text: 'Entertainment',
        id: 'list5',
        child: [
          { id: 'list5_1', text: 'Televisions' },
          { id: 'list5_2', text: 'Home Theatres' },
          { id: 'list5_3', text: 'Gaming Laptops' }
        ]
      }
    ]
      };
    },
    methods: {
      onSelect: function(e) {
          if (e.data && !(e.data.child)) {
              this.$refs.listview.ej2Instances.element.style.display = 'none';
              document.getElementById('close').style.display = 'none';
          }
      },
      onClick: function(e){
          this.$refs.listview.ej2Instances.element.style.display = 'none';
          document.getElementById('close').style.display = 'none';
      },
      hamburgerClick: function() {
          var element = this.$refs.listview.ej2Instances.element;
          var animation = new Animation({ duration: 500 });
          animation.animate(element, {
          name: 'SlideDown',
```

```
        begin: function(args) {
            element.style.display = 'block';
            document.getElementById('close').style.display = 'block';
        }
        });
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
.layoutWrapper {
  line-height: initial;
  border: 1px solid black;
  width: 285px;
  height: 505px;
  margin: auto;
  margin-bottom: 15px;
  border-radius: 28px;
  position: relative;
  background-image: linear-gradient(to top, #ffffff, #f5f5f5);
}
.layoutWrapper .speaker {
  border: 1px solid black;
  border-radius: 5px;
  width: 33.33333333%;
  height: 5px;
  margin: 15px auto 0px auto;
  position: relative;
}
.layoutWrapper .outerButton {
  width: 30px;
  height: 30px;
  border: 1px solid black;
  border-radius: 50%;
  position: absolute;
  bottom: calc(0% + 10px);
  left: calc(50% - 15px);
}
.layoutWrapper .camera {
  position: absolute;
  left: calc(-15% - 10px);
  top: -100%;
  width: 10px;
  height: 10px;
  border-radius: 50%;
  border: 1px solid black;
}
.layoutWrapper .layout {
  border: 1px solid black;
  margin: 20px 13px 0px 13px;
}
.layout #container {
  height: 405px;
  background-color: white;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
#header {
  width: 100%;
  background-color: #7b8cfb;
}
.layout .e-listview .e-list-header,
.layout .e-listview .e-but-back {
  background-color: #7b8cfb;
  color: white;
}
#header .content {
  background-color: #7b8cfb;
  color: white;
  border: none;
  font-size: 20px;
  line-height: 50px;
  text-align: center;
}
#hamburger.menu {
  font-size: 25px;
  cursor: pointer;
  float: left;
  line-height: 50px;
  position: absolute;
  z-index: 1000;
  left: 25px;
  color: white;
}
#hamburger.menu:before {
  content: '\e99a';
}
#listview {
  position: absolute;
  width: 257px;
  overflow: hidden;
  top: 43px;
  left: 14px;
  z-index: 1000;
}
#close:before {
  content: "\e945";
}
#close {
  position: absolute;
  right: 25px;
  color: white;
  top: 60px;
  z-index: 1000;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs24" %}

Copyright © 2001 -2024 Syncfusion Inc.

Accessibility in Vue Menu component

The Menu component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Menu component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | |

| Section 508 Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| Accessibility Checker Validation | |

| Axe-core Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

 - All features of the component meet the requirement.

 - Some features of the component do not meet the requirement.

 - The component does not meet the requirement.

Copyright © 2001 -2024 Syncfusion Inc.

WAI-ARIA attributes

The Menu component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Menu component:

| Attributes | Purpose |

| --- | --- |

| role | Indicates Menu component's root menu as menubar, popup as menu, and the popup items as menuitem. |

| aria-haspopup | Indicates the availability and type of interactive popup element. |

| aria-expanded | Indicates whether the subtree can be expanded or collapsed, and indicates whether its current state can be expanded or collapsed. |

| aria-orientation | Indicates whether the orientation is horizontal or vertical. The default orientation is horizontal. |

| aria-label | Indicates the menu item text. |

| aria-disabled | Indicates the state of menu item whether it is disabled. |

Keyboard interaction

The Menu component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Menu component.

| **Press** | **To do this** |

| --- | --- |

| Esc | Closes the sub menu that contains focus and returns focus to the parent element. |

| Enter | Opens the sub menu if focused menu item has sub menu, and places focus on its first item or activates the item and closes the sub menu. |

| Up | Navigates up or to the previous menu item. |

| Down | Navigates down or to the next menu item. |

| Left | Closes the current sub menu and navigates to the parent menu. |

| Right | Navigates and open the next sub menu. |

| Home | Focuses the first item. |

| End | Focuses the last item.

Ensuring accessibility

The Menu component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Menu component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Menu component with accessibility tools.

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

See also

- Accessibility in Syncfusion Vue components

Style and appearance in Vue Menu component

To modify the Menu appearance, you need to override the default CSS of Menu component. Please find the list of CSS classes and its corresponding section in Menu component. Also, you have an option to create your own custom theme for the controls using our Theme Studio.

CSS Class | Purpose of Class

|.e-menu-wrapper|To customize the menu wrapper

|.e-menu-wrapper.e-menu-popup|To customize the menu popup wrapper

|.e-menu-wrapper ul .e-menu-item|To customize the menu items

|.e-menu-wrapper.e-menu-popup .e-menu-item|To customize the menu popup items

|.e-menu-wrapper ul .e-menu-item .e-caret|To customize the menu items caret icon

How To

Change orientation in Vue Menu component

Orientation in menu items can be changed horizontally or vertically using the orientation property. By default, it is horizontally aligned.

APP.VUE

```
<template>
<div>
<ejs-menu :items='menuItems' orientation='Vertical'></ejs-menu>
</div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
export default {
   data: function() {
        return {
           menuItems:  [
        {
        text: 'File',
        items: [
            { text: 'Open' },
            { text: 'Save' },
            { text: 'Exit' }
        ]
    },
    {
        text: 'Edit',
        items: [
            { text: 'Cut' },
            { text: 'Copy' },
```

```
            { text: 'Paste' }
        ]
    },
    {
        text: 'View',
        items: [
            { text: 'Toolbar' },
            { text: 'Sidebar' },
            { text: 'Full Screen' }
        ]
    },
    {
        text: 'Tools',
        items: [
            { text: 'Spelling & Grammar' },
            { text: 'Customize' },
            { text: 'Options' }
        ]
    },
    { text: 'Help' }
    ]
    };
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs8" %}

Customize menu using events in Vue Menu component

The Menu provides a set of events to enable users to customize it.

APP.VUE

```
<template>
 <div id='container'>
        <div class="control-section">
            <div class="menu-section">
                <ejs-menu :items='menuItems' :beforeOpen='beforeOpen'
:beforeClose='beforeClose' :onClose='onClose' :onOpen='onOpen'
:select='select'></ejs-menu>
            </div>
            <div class="property-section">
                <table id="propertyTable" title="Event trace">
                    <tbody>
                        <th>Event trace:-</th>
                        <tr>
                            <td></td>
```

```
                        </tr>
                    </tbody>
                </table>
            </div>
            <button id='clear' @click=onClick>Clear</button>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
export default {
   data: function() {
        return {
           menuItems:  [
        {
        text: 'Events',
        items: [
            { text: 'Conferences' },
            { text: 'Music' },
            { text: 'Workshops' }
        ]
    },
    {
        text: 'Movies',
        items: [
            { text: 'Now Showing' },
            { text: 'Coming Soon' }
        ]
    },
    {
        text: 'Directory',
        items: [
            { text: 'Media Gallery' },
            { text: 'Newsletters' }
        ]
    },
    {
        text: 'Queries',
        items: [
            { text: 'Our Policy' },
            { text: 'Site Map' }
        ]
    },
    { text: 'Services' }
    ]
    };
    },
    methods: {
updateEventLog: function(args) {
    var propertyElem = document.getElementById('propertyTable');

propertyElem.getElementsByTagName('td')[0].insertAdjacentHTML('beforeend',
args.name + ' Event triggered. <br />');
```

```
},
beforeOpen: function(args) {
    this.updateEventLog(args);
},
beforeClose: function(args) {
    this.updateEventLog(args);
},
onClose: function(args) {
    this.updateEventLog(args);
},
onOpen: function(args) {
    this.updateEventLog(args);
},
select: function(args) {
    this.updateEventLog(args);
},
onClick: function(args) {
    var propertyElem = document.getElementById('propertyTable');
    propertyElem.getElementsByTagName('td')[0].innerHTML = '';
}
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
html, body, .control-section {
    height: 95%;
}
.menu-section {
    margin-top: 100px;
    text-align: center;
    float: left;
}
.property-section {
    overflow: auto;
    width: 40%;
    height: 330px;
    float: right;
    font-family: monospace;
}
.property-section th {
    text-align: left;
}
#clear {
    float: right;
    clear: both;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs14" %}

Customize menu items in Vue Menu component

Add or remove menu items

Menu items can be added or removed using the insertAfter, insertBefore and removeItems methods.

In the following example, the **Europe** menu items are added before the **Oceania** item, the **Africa** menu items are added after the **Asia**, and the **South America** and **Mexico** items are removed from menu.

APP.VUE

```
<template>
<div>
<ejs-menu ref="menu" id="menu" :items='menuItems' :fields='menuFields'
:created='onCreated'></ejs-menu>
</div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
export default {
   data: function() {
        return {
           menuItems:  [
            {
                continent: 'Asia',
                countries: [
                    { country: 'China' },
                    { country: 'India' },
                    { country: 'Japan' }
                ]
            },
            {
                continent: 'North America',
                countries: [
                    { country: 'Canada' },
                    { country: 'Mexico' },
                    { country: 'USA' }
                ]
            },
            {
                continent: 'South America',
                countries: [
                    { country: 'Brazil' },
                    { country: 'Colombia' },
                    { country: 'Argentina' }
                ]
            },
            {
                continent: 'Oceania',
                countries: [
                    { country: 'Australia' },
                    { country: 'New Zealand' },
                    { country: 'Samoa' },
                ]
            },
```

```
            { continent: 'Antarctica' }
        ],
        menuFields: {
            text: ['continent', 'country'],
            children: ['countries']
        }
    };
  },
  methods: {
      onCreated: function(args) {
          var insertItem = [
              {
                  continent: 'Europe',
                  countries: [
                      { country: 'Finland' },
                      { country: 'Austria' }
                  ]
              }
          ];
          this.$refs.menu.insertBefore(insertItem, 'Oceania', false);
          insertItem = [
              {
                  continent: 'Africa',
                  countries: [
                      { country: 'Nigeria' }
                  ]
              }
          ];
          this.$refs.menu.insertAfter(insertItem, 'Oceania', false);
          this.$refs.menu.removeItems(['South America', 'Mexico'], false);
      }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs11" %}

To process items with ID values, set isUnique to true.

Enable or disable menu items

You can enable and disable the menu items using the enableItems method in Menu. To enable menuItems, set the enable property in argument to true and vice-versa.

Install Syncfusion Buttons packages using below command.

`bash

npm install @syncfusion/ej2-vue-buttons --save

`

In the following example, the **Directory** header item, **Conferences**, and **Music** sub menu items are disabled.

APP.VUE

```
<template>
    <div>
        <ejs-button ref='enableAll'  v-on:click.native="btnClick">Enable all items</ejs-button>
        <div class="menu-section">
            <ejs-menu ref="menu" id="menu" :items='menuItems' :created='onCreated' :beforeOpen='beforeOpen'></ejs-menu>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
Vue.use(ButtonPlugin);
var disableItems = ['Conferences', 'Music', 'Directory'];
export default {
   data: function() {
        return {
           menuItems:  [
    {
        text: 'Events',
        items: [
            { text: 'Conferences' },
            { text: 'Music' },
            { text: 'Workshops' }
        ]
    },
    {
        text: 'Movies',
        items: [
            { text: 'Now Showing' },
            { text: 'Coming Soon' }
        ]
    },
    {
        text: 'Directory',
        items: [
            { text: 'Media Gallery' },
            { text: 'Newsletters' }
        ]
    },
    {
        text: 'Queries',
        items: [
            { text: 'Our Policy' },
```

```
            { text: 'Site Map' }
        ]
    },
    { text: 'Services' }
]
     };
   },
   methods: {
       beforeOpen: function(args) {
        for (var i = 0; i  < args.items.length; i++) {
            if (disableItems.indexOf(args.items[i].text) > -1) {
                this.$refs.menu.enableItems([args.items[i].text], false,
false);
            }
        }
       },
       btnClick: function(event) {
           this.$refs.menu.enableItems(disableItems, true, false);
           disableItems = [];
       },
       onCreated: function(args) {
           this.$refs.menu.enableItems(disableItems, false, false);
       }
   }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
.menu-section {
    margin-top: 100px;
    text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs12" %}

To disable sub menu items, use the beforeOpen event.

Hide or show menu items

You can show or hide the menu items using the showItems and hideItems methods.

APP.VUE

```
<template>
    <div>
        <ejs-button ref='enableAll' v-on:click.native="btnClick">Show all
items</ejs-button>
        <div class="menu-section">
            <ejs-menu ref="menu" id="menu" :items='menuItems'
:created='onCreated' :beforeOpen='beforeOpen'></ejs-menu>
        </div>
    </div>
</template>
<script>
```

```
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
Vue.use(ButtonPlugin);
var hiddenItems = ['Workshops', 'Music', 'Movies'];
export default {
   data: function() {
        return {
           menuItems:  [
    {
        text: 'Events',
        items: [
            { text: 'Conferences' },
            { text: 'Music' },
            { text: 'Workshops' }
        ]
    },
    {
        text: 'Movies',
        items: [
            { text: 'Now Showing' },
            { text: 'Coming Soon' }
        ]
    },
    {
        text: 'Directory',
        items: [
            { text: 'Media Gallery' },
            { text: 'Newsletters' }
        ]
    },
    {
        text: 'Queries',
        items: [
            { text: 'Our Policy' },
            { text: 'Site Map' }
        ]
    },
    { text: 'Services' }
]
     };
   },
   methods: {
       beforeOpen: function(args) {
        for (var i = 0; i  < args.items.length; i++) {
            if (hiddenItems.indexOf(args.items[i].text) > -1) {
                this.$refs.menu.hideItems([args.items[i].text], false);
            }
        }
       },
       btnClick: function(event) {
           this.$refs.menu.showItems(hiddenItems, false);
           hiddenItems = [];
       },
```

```
        onCreated: function(args) {
            this.$refs.menu.hideItems(hiddenItems, false);
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
.menu-section {
    margin-top: 100px;
    text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs13" %}

Using the beforeOpen event, you can hide the sub menu items as in the above example since the menu supports to hide items only for headers initially.

Create mnemonic ui in menuitem in Vue Menu component

A particular character in a text can be underlined in the beforeItemRender event by adding `<u>` tag in between the text and assign the innerHTML to the `li` element.

In the following example, the first character in `File`, `Open`, and `Save` menu items are underlined.

APP.VUE

```
<template>
    <div>
        <ejs-menu ref="menu" id="menu" :items='menuItems'
:beforeItemRender='beforeItemRender'></ejs-menu>
    </div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
export default {
   data: function() {
        return {
           menuItems:   [
        {
            text: 'File',
            iconCss: 'em-icons e-file',
            items: [
                { text: 'Open', iconCss: 'em-icons e-open' },
                { text: 'Save', iconCss: 'e-icons e-save' },
                { separator: true },
                { text: 'Exit' }
            ]
        },
        {
```

```
            text: 'Edit',
            iconCss: 'em-icons e-edit',
            items: [
                { text: 'Cut', iconCss: 'em-icons e-cut' },
                { text: 'Copy', iconCss: 'em-icons e-copy' },
                { text: 'Paste', iconCss: 'em-icons e-paste' }
            ]
        },
        { text: 'Format' },
        { text: 'View' },
        { text: 'Bookmarks' },
        { text: 'Tools' },
        { separator: true },
        { text: 'Help' }
    ]
     };
   },
   methods: {
       beforeItemRender: function(args) {
         if (['File', 'Open', 'Save'].indexOf(args.item.text) > -1) {
            // To underline a First character.
            var underlinedText = '<u>' + args.item.text.slice(0, 1) + '</u>'
+ args.item.text.slice(1);
            args.element.innerHTML =
args.element.innerHTML.replace(args.item.text, underlinedText);
          }
       }
   }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
/**
* ej2 Menu styles
*/
@font-face {
    font-family: 'e-menu';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMjvJSpgAAAEoAAAAVmNtYXBsm2feAAABpAAAAGxnbHlmmEc
yrQAAAiQAAAWIaGVhZBJ0bwcAAADQAAAANmhoZWEHmQNyAAAArAAAACRobXR4I0AAAAAAAYAAAAA
kbG9jYQaGB+4AAAIQAAAAFG1heHABGACaAAABCAAAACBuYW1lc0cOBgAAB6wAAAIlcG9zdJbKd4k
AAAnUAAAAfQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAACQABAAAAAQAAhka7o18
PPPUACwPoAAAAANe2FRwAAAAA17YVHAAAAAAD6gPqAAAACAACAAAAAAAAAAAEAAAAJAI4ABQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPrAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5anohgNS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABYAAAADgA
IAAIABuWp5bPluefo6CLohv//AADlqeWy5bnn6Ogi6IX//wAAAAAAAAAAAAAAAAAABAA4ADgAQABA
AEAAQAAAABQAGAAcACAABAAIAAwAEAAAAAACsASoBRAGwAhICUAKEAsQABAAAAAAD6gNZAD8AfwC
DAI0AAAEzHw0dAQ8OLw8/DiMzHw0dAQ8OLw8/DgMhAyEBIRU3EyUVBQMjAwgJCAgIBwcHBgUFBAQ
DAgEBAgMEBAUFBgcHBwgICAkJCAgIBwcHBgUFBAQDAgEBAQECAwQEBQUGBwcHCAgI5AgJCAgHBwc
```

```
GBQUEBAIDAQEDAgQEBQUGBwcHCAgJCAkICAgIBwYGBQUFAwMCAQEBAQIDAwUFBQYGBwgICAijAny
Q/qj+EgEKAssBcf5Yy9UBTwICAgQEBQYGBgcHCAgJCAkICQcIBwYGBgQFAwMCAQEBAQIDAwUEBgY
GBwgHCQgJCAkICAcHBgYGBQQEAgICAgICBAQFBgYGBwcICAkICQgJBwgHBgYGBAUDAwIBAQEBAgM
DBQQGBgYHCAcJCAkICQgIBwcGBgYFBAQCAgIBu/67AZUBAf5LAj0CAbUAAAAFAAAAAAPqA+oAAgA
WABgAPABkAAA3OQEnMx8PFQc3MQE7AR8OAQcvDwEzHwoPBi8PPwP/nAgODg4NDAwLCwoICAcFBAM
C6k4CdAgHEA4PDQ0MDAoJCAcGBAIB/kWFAQMEBgcJCgsLDQ0NDg4ODgLaBg0GBgYGBjwFBAMBAQE
CAgYJNAEDBAYHCQoKDAwNDQ4ODg40GQkKZJsBAwQFBwcJCQoMCw0NDg8OCE7pAnUDBQYHCQkLDAw
NDg0ODg7+SIgODg4NDg0MDAsKCAgGBAMBArUCAgMDBQU9CQkJCQgICAcNDjQNDg4ODQ0MDAsJCQc
GBAMBNA4DAgAAAAABAAAAAAPqA60ACgAAEyEVIRUhAxMhAyEVAcwBzPzEN5MDHrj84gOtXFz9/QG
n/boAAAAABQAAAAADjgPqAAMABwALAA8AUwAAEyEVITUhFSE1IRUhJxEhESUhHw8RDw8hLw8RPw7
qAij92AIo/dgCKP3YOwKi/XICeggICAgHBwYGBQUEAwMCAQEBAQIDAwQFBQYGBwcICAgI/YYICAg
IBwcGBgUFBAMDAgEBAQECAwMEBQUGBgcHCAgIAQs+9j72Prj9XgKi9gEBAgMDBAUFBgYHBwgICAj
8zggICAgHBwYGBQUEAwMCAQEBAQIDAwQFBQYGBwcICAgIAzIICAgIBwcGBgUFBAMDAgEABQAAAAA
DqQOpAAQACgAUAB4AOwAACQEXATUBFAcmNDIDBgcuATQ2MhYUAwYHLgE0NjIWFBc2NS4BIgYUFhc
yNxcHJiMOARQWMjY3NCc3ATM1Ayb++FkBMv5fFRUq3xglJjExSzEZGCUmMTFLMUoOAmKUY2JLJyF
mZiEnS2JilWICDmcBM4MDgP74WQE2K/50FQICKv6lGQICMkoyMkoB9xkCAjJKMjJKIyEnSmNjlGM
CDmdnDgJjlGNjSichZ/7NKwAAAAMAAAAAA4oD5gAHABAAJwAAARUhNTMRIRElHgEGIiY0NjInBgc
jIgYVERQWMyEyNjURNCYrAS4BIgEZAbZd/ZABWAwBGiYZGSZhIg+8JjU1JgJ2JjU1JrwPRFgDLn5
9/TICz1IMJxkZJxlAGSkzJv0pJzMzJwLXJjMpMwADAAAAAAOpA+cAAwAUAB4AAAERIREnBhURFBY
XIT4BNRE0JiMhIicGFREzESE1IQYDTP4MQxs2JgH3JzU1J/4JJtgZXQIT/egmAs/9jwJxRBkm/Yc
mMwICMyYCeSYynxon/Y8CcV4CAAIAAAAAA+cD5wALACMAAAEOAQcuASc+ATceAQUeARcyNj8BARY
yNjQnATc+ATUuAScOAQLYA7SHiLMEBLOIh7T9KwXnrkeBNAMBAQ4kHA7+/wMpLgTora7nAk6HtAM
DtIeIswQEs4it6AQuKQP+/w4bJQ4BAQM0gUeu5wUF5wAAAAASAN4AAQAAAAAAAABAAAAAQAAAAA
AAQAHAAEAAQAAAAAAgAHAAgAAQAAAAAAwAHAA8AAQAAAAAABAAHABYAAQAAAAAABQALAB0AAQA
AAAAABgAHACgAAQAAAAAACgAsAC8AAQAAAAAACwASAFsAAwABBAkAAAACAG0AAwABBAkAAQAOAG8
AAwABBAkAAgAOAH0AAwABBAkAAwAOAIsAAwABBAkABAAOAJkAAwABBAkABQAWAKcAAwABBAkABgA
OAL0AAwABBAkACgBYAMsAAwABBAkACwAkASMgZS1pY29uc1JlZ3VsYXJlLWljb25zZS1pY29uc1Z
lcnNpb24gMS4wZS1pY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R
1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGkAYwBvAG4AcwBSAGUAZwB1AGwAYQByAGUALQB
pAGMAbwBuAHMAZQAtAGkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAZQAtAGkAYwBvAG4
AcwBGAG8AbgB0ACAAZwBlAG4AZQByAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQB
zAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGk
AbwBuAC4AYwBvAG0AAAAAAgAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAJAQIBAwEEAQU
BBgEHAQgBCQEKAA1zaG9wcGluZy1jYXJ0B2VkaXQtMDUMZmlsZS1vcGVuLTAxDGZpbGUtdGV4dC0
wMQNDdXQFUGFzdGUEQ29weQZTZWFyY2gAAAAAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.em-icons {
    font-family: 'e-menu';
    font-style: normal;
    font-variant: normal;
    font-weight: normal;
    line-height: 1;
    text-transform: none;
}
.e-file::before {
    content: '\e886';
}
.e-edit::before {
    content: '\e822';
}
.e-cut::before {
    content: '\e5a9';
}
.e-copy::before {
    content: '\e5b3';
```

```
}
.e-paste::before {
    content: '\e5b2';
}
.e-open::before {
    content: '\e885';
}
.e-save::before {
    content: '\e98e';
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs10" %}

Change sub menu position in Vue Menu component

The submenu position can be changed by using the beforeOpen event. Assign the top and left position where you want to open the submenu to the beforeOpen event arguments args.top and args.left respectively.

In the below sample, the sub menu opens above the parent menu item.

APP.VUE

```
<template>
    <div>
        <ejs-menu :items='menuItems' :beforeOpen='onBeforeOpen'></ejs-menu>
    </div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple, closest } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
export default {
  data: function() {
    return {
        // Menu items definition
        menuItems: [
        {
            text: 'File',
            items: [
                { text: 'Open' },
                { text: 'Save' },
                { text: 'Exit' }]
        },
        {
            text: 'Edit',
            items: [
                { text: 'Cut' },
                { text: 'Copy' },
                { text: 'Paste' }]
        },
        {
            text: 'View',
            items: [
```

```
                { text: 'Toolbar' },
                { text: 'Sidebar' }]
        },
        {
            text: 'Tools',
            items: [
                { text: 'Spelling & Grammar' },
                { text: 'Customize' },
                { text: 'Options' }]
        },
        { text: 'Go' },
        { text: 'Help' }
    ]
    };
  },
   methods: {
        onBeforeOpen(args) {
            // Getting parent menu item element offset
            let relativeOffset = closest(args.event.target, '.e-menu-
item').getBoundingClientRect();
            // Getting sub menu wrapper element using closest method
            let subMenuEle = closest(args.element, '.e-menu-wrapper');
            subMenuEle.style.display = 'block';
            args.top = (relativeOffset.top -
subMenuEle.getBoundingClientRect().height) + pageYOffset;
            args.left = relativeOffset.left + pageXOffset;
            subMenuEle.style.display = '';
        }
   }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 200px;
    text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs9" %}

> For custom positioning, set both `top` and `left` position in the beforeOpen event.

Set title in Vue Menu component

In this sample , the title for menu item can be achievable by using 'beforeItemRender' client-side event in Menu component.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="menu-section">
            <ejs-menu :items='menuItems'
:beforeItemRender='beforeItemRender' ></ejs-menu>
```

```
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
export default {
   data: function() {
        return {
           menuItems:  [
        {
            id: 'settingIcon',
            iconCss: 'em-icons e-file',
            items: [
                { text: 'Open',
                  items: [
                      { text: 'Sub Option1' },
                      { text: 'Sub Option2' },
                  ]
                },
                { text: 'Save' },
                { separator: true },
                { text: 'Exit' }
            ]
        }
    ]
    };
    },
    methods: {
beforeItemRender: function(args) {
    if (args.item.id == 'settingIcon') {
        args.element.setAttribute('title', 'Settings');
      }
},
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
body {
    margin-top: 100px;
    text-align: center;
}
.em-icons {
    font-family: 'e-menu';
    font-style: normal;
    font-variant: normal;
    font-weight: normal;
    line-height: 1;
    text-transform: none;
}
.e-file::before {
```

```
    content: '\e886';
}
.e-menu-wrapper .e-menu .e-menu-item.e-menu-caret-icon .e-icons.e-caret {
    display: none;
  }

  .e-menu-wrapper .e-menu .e-menu-item.e-menu-caret-icon {
    padding-right: 5px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs17" %}

Rounded corner in Vue Menu component

The rounded corner can be achieved by using the cssClass property. Add a custom class to the menu component and customize it using the border-radius CSS property. For more information, refer to the styles specified in the below sample.

APP.VUE

```
<template>
    <div>
        <ejs-menu :items='menuItems' cssClass='e-rounded-menu'></ejs-menu>
    </div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(MenuPlugin);
export default {
  data: function() {
    return {
        // Menu items definition
        menuItems: [
        {
            text: 'File',
            items: [
                { text: 'Open' },
                { text: 'Save' },
                { text: 'Exit' }]
        },
        {
            text: 'Edit',
            items: [
                { text: 'Cut' },
                { text: 'Copy' },
                { text: 'Paste' }]
        },
        {
            text: 'View',
            items: [
                { text: 'Toolbar' },
                { text: 'Sidebar' }]
        },
        {
```

```
            text: 'Tools',
            items: [
                { text: 'Spelling & Grammar' },
                { text: 'Customize' },
                { text: 'Options' }]
        },
        { text: 'Go' },
        { text: 'Help' }
    ]
    };
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/bootstrap.css";
body {
    margin-top: 100px;
    text-align: center;
}
/* Styles to achieve rounder corner in menu */
.e-menu-wrapper.e-rounded-menu:not(.e-menu-popup),
.e-menu-wrapper.e-rounded-menu .e-menu {
  border-radius: 20px;
}
/* Increased the menu component left and right padding for better rounded
corner UI */
.e-menu-wrapper.e-rounded-menu ul.e-menu {
  padding: 0 14px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs16" %}

Menu item click in Vue Menu component

You can open menu items and sub menu on menu item click by setting showItemOnClick property of the Menu. To open sub menu items only on item click, should be set as true.

APP.VUE

```
<template>
    <div>
        <ejs-menu :items='menuItems' showItemOnClick = "true"></ejs-menu>
    </div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(MenuPlugin);
export default {
  data: function() {
    return {
        // Menu items definition
        menuItems: [
        {
```

```
            text: 'File',
            items: [
                { text: 'Open' },
                { text: 'Save' },
                { text: 'Exit' }]
        },
        {
            text: 'Edit',
            items: [
                { text: 'Cut' },
                { text: 'Copy' },
                { text: 'Paste' }]
        },
        {
             text: 'View',
             items: [
                 {
                 text: 'Toolbars',
                 items: [
                     { text: 'Menu Bar' },
                     { text: 'Bookmarks Toolbar' },
                     { text: 'Customize' },
                   ]
                 },
                {
                 text: 'Zoom',
                 items: [
                     { text: 'Zoom In' },
                     { text: 'Zoom Out' },
                     { text: 'Reset' },
                    ]
                },
                { text: 'Full Screen' }
            ]
        },
        {
            text: 'Tools',
            items: [
                { text: 'Spelling & Grammar' },
                { text: 'Customize' },
                { text: 'Options' }]
        },
        { text: 'Go' },
        { text: 'Help' }
    ]
    };
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/bootstrap.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/bootstrap.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/bootstrap.css";
body {
    margin-top: 100px;
    text-align: center;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
/* Styles to achieve rounder corner in menu */
.e-menu-wrapper.e-rounded-menu:not(.e-menu-popup),
.e-menu-wrapper.e-rounded-menu .e-menu {
  border-radius: 20px;
}
/* Increased the menu component left and right padding for better rounded
corner UI */
.e-menu-wrapper.e-rounded-menu ul.e-menu {
  padding: 0 14px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/menu/getting-started-cs15" %}

Message

Getting Started with the Vue Message Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Message component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

vue create quickstart

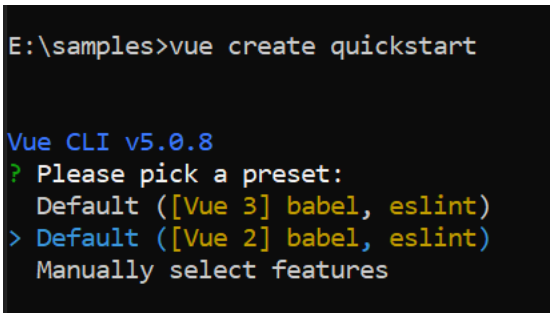
cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.

Copyright © 2001 -2024 Syncfusion Inc.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Message component as an example. Install the @syncfusion/ej2-vue-notifications package by running the following command:

`bash

npm install @syncfusion/ej2-vue-notifications --save

`

or

`bash

yarn add @syncfusion/ej2-vue-notifications

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, [CRG] and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Message component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Message component using Composition API or Options API:

1\. First, import and register the Message component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { MessageComponent as EjsMessage } from "@syncfusion/ej2-vue-
notifications";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { MessageComponent } from "@syncfusion/ej2-vue-notifications";
//Component registration
export default {
name: 'app',
components: {
'ejs-message': MessageComponent
}
}
</script>
```

2\. In the template section, define the Message component with the content property.

~/SRC/APP.VUE

```
<template>
<ejs-message id="msg" :content="content"></ejs-message>
</template>
```

3\. Declare the value for the content property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const content = "Please read the comments carefully";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
content: "Please read the comments carefully"
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  <ejs-message id="msg" :content="content"></ejs-message>
</template>
<style>
  .e-message {
    margin: 100px;
  }
</style>
<script setup>
import { MessageComponent as EjsMessage } from "@syncfusion/ej2-vue-
notifications";
const content = 'Please read the comments carefully';
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <ejs-message id="msg" :content="content"></ejs-message>
</template>
<style>
  .e-message {
    margin: 100px;
  }
</style>
<script>
import { MessageComponent } from "@syncfusion/ej2-vue-notifications";
export default {
  name: "app",
  components: {
    'ejs-message': MessageComponent
  },
  data() {
    return {
      content: "Please read the comments carefully"
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/message/getting-started-cs4" %}

Getting Started with the Vue Message Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Message component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Copyright © 2001 -2024 Syncfusion Inc.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Message component as an example. To use the Vue Message component in the project, the @syncfusion/ej2-vue-notifications package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-notifications --save

`

or

`bash

yarn add @syncfusion/ej2-vue-notifications

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Message component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue Message component using Composition API or Options API:

1.First, import and register the Message component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { MessageComponent as EjsMessage } from "@syncfusion/ej2-vue-
notifications";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { MessageComponent } from '@syncfusion/ej2-vue-notifications';
//Component registration
export default {
name: "App",
components: {
'ejs-message': MessageComponent
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
```

2.In the template section, define the Message component with the content property.

~/SRC/APP.VUE

```
<template>
<ejs-message id="msg" :content="content"></ejs-message>
</template>
```

3.Declare the value for the content property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const content = "Please read the comments carefully";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
content: "Please read the comments carefully"
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-message id="msg" :content="content"></ejs-message>
</template>
<script setup>
import { MessageComponent as EjsMessage } from "@syncfusion/ej2-vue-
notifications";
const content = "Please read the comments carefully";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-message id="msg" :content="content"></ejs-message>
</template>
<script>
import { MessageComponent } from '@syncfusion/ej2-vue-notifications';
//Component registration
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
name: "App",
components: {
'ejs-message': MessageComponent
},
data() {
return {
content: "Please read the comments carefully"
};
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

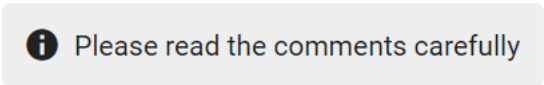
or

`bash

yarn run dev

`

The output will appear as follows:



For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Severities in Vue Message component

The severity denotes the importance and context of the message to the user. The message contains different severity types. Use the severity property to display the messages with different severity levels.

The available severity types are **Normal**, **Success**, **Info**, **Warning** and **Error**. The default severity type for messages is **Normal**.

The following example demonstrates the severity of the messages.

APP.VUE

```
<template>
  <div class="msg-default-section">
    <div class="content-section">
      <ejs-message id="msg_default" content="Editing is restricted"></ejs-
message>
      <ejs-message id="msg_info" content="Please read the comments
carefully" severity="Info"></ejs-message>
      <ejs-message id="msg_success" content="Your message has been sent
successfully" severity="Success"></ejs-message>
      <ejs-message id="msg_warning" content="There was a problem with your
network connection" severity="Warning"></ejs-message>
      <ejs-message id="msg_error" content="A problem occurred while
submitting your data" severity="Error"></ejs-message>
    </div>
  </div>
</template>
<script setup>
import { MessageComponent as EjsMessage } from "@syncfusion/ej2-vue-
notifications";
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
  .msg-default-section .content-section {
    margin: 0 auto;
    max-width: 450px;
    padding-top: 10px;
  }
  .msg-default-section .e-message {
    margin: 10px 0;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/message/getting-started-cs8" %}

Variants in Vue Message component

The Message has predefined appearance variants for different visual representations. The variants of the message can be changed based on the variant property.

The available variants are **Text**, **Outlined** and **Filled**. The default variant type for messages is **Text**.

- **Text** - The severity is differentiated using a text color and a light background color.
- **Outlined** - The severity is differentiated using a text color and a border without a background.
- **Filled** - The severity is differentiated using a text color and a dark background color.

The following example demonstrates the default message with different variant types.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <div class="msg-variant-section">
    <div class="content-section">
      <h4>Filled</h4>
      <ejs-message id="msg_default_filled" variant="Filled">Editing is
restricted</ejs-message>
      <ejs-message id="msg_info_filled" severity="Info"
variant="Filled">Please read the comments carefully</ejs-message>
      <ejs-message id="msg_success_filled" severity="Success"
variant="Filled">Your message has been sent successfully</ejs-message>
      <ejs-message id="msg_warning_filled" severity="Warning"
variant="Filled">There was a problem with your network connection</ejs-
message>
      <ejs-message id="msg_error_filled" severity="Error" variant="Filled">A
problem occurred while submitting your data</ejs-message>
    </div>
    <div class="content-section">
      <h4>Outlined</h4>
      <ejs-message id="msg_default_outlined" variant="Outlined">Editing is
restricted</ejs-message>
      <ejs-message id="msg_info_outlined" severity="Info"
variant="Outlined">Please read the comments carefully</ejs-message>
      <ejs-message id="msg_success_outlined" severity="Success"
variant="Outlined">Your message has been sent successfully</ejs-message>
      <ejs-message id="msg_warning_outlined" severity="Warning"
variant="Outlined">There was a problem with your network connection</ejs-
message>
      <ejs-message id="msg_error_outlined" severity="Error"
variant="Outlined">A problem occurred while submitting your data</ejs-
message>
    </div>
    <div class="content-section">
      <h4>Text</h4>
      <ejs-message id="msg_default">Editing is restricted</ejs-message>
      <ejs-message id="msg_info" severity="Info">Please read the comments
carefully</ejs-message>
      <ejs-message id="msg_success" severity="Success">Your message has been
sent successfully</ejs-message>
      <ejs-message id="msg_warning" severity="Warning">There was a problem
with your network connection</ejs-message>
      <ejs-message id="msg_error" severity="Error">A problem occurred while
submitting your data</ejs-message>
    </div>
  </div>
</template>
<script setup>
import { MessageComponent as EjsMessage } from "@syncfusion/ej2-vue-
notifications";
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
.msg-variant-section .content-section {
  margin: 0 auto;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  max-width: 520px;
  padding: 10px;
}
.msg-variant-section .e-message {
  margin: 10px 0;
}
.msg-variant-section {
  display: flex;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/message/getting-started-cs10" %}

Icons in Vue Message component

This section explains the message with no icons, how to show or hide the close icon and add the custom severity icon to the message.

No Icon

By default, severity icons can be displayed according to the severity types to make it more understandable to the user by visual information rather than text. To hide the severity icons, set the `showIcon` property to `false`.

The following example demonstrates the different severity messages without the severity icons.

APP.VUE

```
<template>
  <div class="msg-default-section">
    <div class="content-section">
      <ejs-message id="msg_default" content="Editing is restricted"
:showIcon="showIcon"></ejs-message>
      <ejs-message id="msg_info" content="Please read the comments
carefully" severity="Info" :showIcon="showIcon"></ejs-message>
      <ejs-message id="msg_success" content="Your message has been sent
successfully" severity="Success" :showIcon="showIcon"></ejs-message>
      <ejs-message id="msg_warning" content="There was a problem with your
network connection" severity="Warning" :showIcon="showIcon"></ejs-message>
      <ejs-message id="msg_error" content="A problem occurred while
submitting your data" severity="Error" :showIcon="showIcon"></ejs-message>
    </div>
  </div>
</template>
<script setup>
import { MessageComponent as EjsMessage } from "@syncfusion/ej2-vue-
notifications";
const showIcon = true;
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
  .msg-default-section .content-section {
    margin: 0 auto;
    max-width: 450px;
    padding-top: 10px;
  }
```

```
  .msg-default-section .e-message {
    margin: 10px 0;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/message/getting-started-cs5" %}

Custom Icon

By default, the severity icons can be displayed according to the severity type to make it more understandable to the user by visual information rather than text. If the user wants to customize these icons, it can be achieved through the cssClass property.

The following example demonstrates how the default message is rendered with a custom severity icon.

APP.VUE

```
<template>
  <div class="col-lg-12 control-section msg-custom-section">
    <div class="content-section">
      <ejs-message id="msg_icon" cssClass="custom">Essential JS 2 is a
modern JavaScript UI Controls library that has been built from the ground up
to be lightweight, responsive, modular and touch friendly. It is written in
TypeScript and has no external dependencies. It also includes complete
support for Angular, React, Vue, ASP.NET MVC and ASP.NET Core frameworks.
      </ejs-message>
    </div>
  </div>
</template>
<script setup>
import { MessageComponent as EjsMessage } from "@syncfusion/ej2-vue-
notifications";
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
  .msg-custom-section .content-section {
    margin: 0 auto;
    max-width: 400px;
    padding-top: 10px;
  }
  .msg-custom-section .e-message {
    margin: 10px 0;
  }
  @font-face {
    font-family: 'Message_icons';
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj8BS4YAAAEoAAAAVmNtYXDopOjpAAABiAAAADZnbHlmEsN
XoQAAAcgAAAJMaGVhZCGnOH4AAADQAAAANmhoZWEIPwQDAAAArAAAACRobXR4CAAAAAAAAYAAAAA
IbG9jYQEmAAAAAAHAAAAABm1heHABEAEOAAABCAAAACBuYW1l1dofAwAABBQAAAJtcG9zdNGGZXA
AAAaEAAAALwABAAAEAAAAAFwEAAAAAAD4gABAAAAAAAAAAAAAAAAAAAAAgABAAAAAQAAecv2N18
PPPUACwQAAAAAAN9TeeEAAAAA31N54QAAAAAD4gO/AAAACAACAAAAAAAAAAEAAAACAQIABAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA6JTolAQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA6JT//wAA6JT//wAAAAEABAAAAAAEAAAA
AAAABJgAAAAQAAAAAA+IDvwANAG4AoAEBAAABLwMrATUzPwQXDwMVHxIdAQ8SFR8GMz8RPQEvESM
```

```
PAicHIw8HERUfDD8JES8JIw8BNw8DFR8SHQEPEhUfBj8SPQEvESMPAgHCjwYHCAelpQcIBwaPqQM
CAwECAwUGCQkIBwcHBQYEBQMDAgIBAQICAwMFBAYFBwcHCAkJBgUDAgEDBQcICQkHBgcGDQwLCwo
JCAcHBgUEAwICAgIDBAUGBwcICQoLCwwNBgcHBwkJB6LJvwkIBwYFBAIBAwQFBgcECMTJBgcHCAg
JBQUEBAMCAQEBAQIDBAQFBQkJBwcH/QMCAwECAwUGFRQSEREODgwLCggGBgMDAwMGBggKCwwODhE
REhQVBgUDAgEDBQcICQkHBgcGGBgWFBMSDw8NCwkIBgUDAwUGCAkLDQ8PEhMUFhgYBgcHBwkJBwE
lfwQEAqgBAgMFfwQDBAgICAgHBwYHCAkJCQkKCgsLCgwLDAsMDAsMCwwKCwsKCgkJCQkIBwYHBwg
ICAgHBwUDAQIDBAoMDAwNDQ4PDg8QEBAQEBEREBAQEBAPDg8ODQ0MDAwKBAMCAQMFerMBAgQFBgc
ECP7/CAgHBgYDAgEBsgUDAgEBAwMEBAUFBQYGAnYGBgYFBQQEAwIBAgROBAMICAgIBwcGERMTFRU
WFxcYGRkaGhsbGxsbGxoaGRkYFxcWFRUTExEGBwcICAgIBwcFAwEBAQMEFRUXGBkaGxsdHR4eHx8
gICAgHx8eHh0dGxsaGRgXFRUEAwIBAwUAAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAEADQABAAE
AAAAAAAIABwAOAAEAAAAAAAMADQAVAAEAAAAAAAQADQAiAAEAAAAAAAUACwAvAAEAAAAAAAYADQA
6AAEAAAAAAAoALABHAAEAAAAAAsAEgBzAAMAAQQJAAAAAgCFAAMAAQQJAAEAGgCHAAMAAQQJAAI
ADgChAAMAAQQJAAMAGgCvAAMAAQQJAAQAGgDJAAMAAQQJAAUAFgDjAAMAAQQJAAYAGgD5AAMAAQQ
JAAoAWAETAAMAAQQJAAsAJAFrIE1lc3NhZ2VfaWNvbnNSZWd1bGFyTWVzc2FnZV9pY29uc01lc3N
hZ2VfaWNvbnNWZXJzaW9uIDEuME1lc3NhZ2VfaWNvbnNGb250IGdlbmVyYXRlZCB1c2luZyBTeW5
jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAE0AZQBzAHMAYQBnAGUAXwB
pAGMAbwBuAHMAUgBlAGcAdQBsAGEAcgBNAGUAcwBzAGEAZwBlAF8AaQBjAG8AbgBzAE0AZQBzAHM
AYQBnAGUAXwBpAGMAbwBuAHMAVgBlAHIAcwBpAG8AbgAgADEALgAwAE0AZQBzAHMAYQBnAGUAXwB
pAGMAbwBuAHMARgBvAG4AdAAgAGcAZQBuAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4
AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB5AG4AYwB
mAHUAcwBpAG8AbgAuAGMAbwBtAAAAAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgE
CAQMABWF1ZGlvAAAA) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  .custom.e-message .e-msg-icon::before {
    font-family: 'Message_icons';
    content: '\e894';
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/message/getting-started-cs6" %}

Close Icon

The message can be rendered with or without the close icon. The close icon is used to hide the message, either by manually clicking the close icon or through keyboard interaction.

By default, the close icon is not rendered in the message. To show the close icon, set the `showCloseIcon` property to `true`.

In the following example, the messages are rendered with the close icon.

APP.VUE

```
<template>
  <div class="col-lg-8 control-section msg-icon-section">
    <div class="content-section">
      <ejs-button id="btn1" ref="defaultBtn" content="Show Default Message"
cssClass="e-outline e-primary msg-hidden"
        v-on:click="defaultClick"></ejs-button>
      <ejs-message id="msg_default_icon" ref="msgDefault"
:showCloseIcon="showCloseIcon" :closed="defaultClosed">Editing
        is restricted</ejs-message>
      <ejs-button id="btn2" ref="infoBtn" content="Show Info Message"
cssClass="e-outline e-primary e-info msg-hidden"
        v-on:click="infoClick"></ejs-button>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      <ejs-message id="msg_info_icon" ref="msgInfo" severity="Info"
:showCloseIcon="showCloseIcon"
        :closed="infoClosed">Please read the comments carefully</ejs-
message>
      <ejs-button id="btn3" ref="successBtn" content="Show Success Message"
        cssClass="e-outline e-primary e-success msg-hidden" v-
on:click="successClick"></ejs-button>
      <ejs-message id="msg_success_icon" ref="msgSuccess" severity="Success"
:showCloseIcon="showCloseIcon"
        :closed="successClosed">Your message has been sent
successfully</ejs-message>
      <ejs-button id="btn4" ref="warningBtn" content="Show Warning Message"
        cssClass="e-outline e-primary e-warning msg-hidden" v-
on:click="warningClick"></ejs-button>
      <ejs-message id="msg_warning_icon" ref="msgWarning" severity="Warning"
:showCloseIcon="showCloseIcon"
        :closed="warningClosed">There was a problem with your network
connection</ejs-message>
      <ejs-button id="btn5" ref="errorBtn" content="Show Error Message"
cssClass="e-outline e-primary e-error msg-hidden"
        v-on:click="errorClick"></ejs-button>
      <ejs-message id="msg_error_icon" ref="msgError" severity="Error"
:showCloseIcon="showCloseIcon"
        :closed="errorClosed">A problem has been occurred while submitting
your data</ejs-message>
    </div>
  </div>
</template>
<script setup>
import { MessageComponent as EjsMessage } from "@syncfusion/ej2-vue-
notifications";
import { ButtonComponent as EjsButton } from "@syncfusion/ej2-vue-buttons";
const showCloseIcon = true;
const defaultClick = function () {
  this.show(this.$refs.msgDefault, this.$refs.defaultBtn);
}
const infoClick = function () {
  this.show(this.$refs.msgInfo, this.$refs.infoBtn);
}
const successClick = function () {
  this.show(this.$refs.msgSuccess, this.$refs.successBtn);
}
const warningClick = function () {
  this.show(this.$refs.msgWarning, this.$refs.warningBtn);
}
const errorClick = function () {
  this.show(this.$refs.msgError, this.$refs.errorBtn);
}
const defaultClosed = function () {
  this.$refs.defaultBtn.ej2Instances.element.classList.remove("msg-hidden");
}
const infoClosed = function () {
  this.$refs.infoBtn.ej2Instances.element.classList.remove("msg-hidden");
}
const warningClosed = function () {
  this.$refs.warningBtn.ej2Instances.element.classList.remove("msg-hidden");
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
const successClosed = function () {
  this.$refs.successBtn.ej2Instances.element.classList.remove("msg-hidden");
}
const errorClosed = function () {
  this.$refs.errorBtn.ej2Instances.element.classList.remove("msg-hidden");
}
const show = function (message, btn) {
  message.ej2Instances.visible = true;
  btn.ej2Instances.element.classList.add("msg-hidden");
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
buttons/styles/button/material.css";
  .msg-icon-section .content-section {
    margin: 0 auto;
    max-width: 450px;
    padding-top: 10px;
  }
  .msg-icon-section .e-message {
    margin: 10px 0;
  }
  .msg-icon-section .e-btn {
    display: block;
    margin: 10px 0;
  }
  .msg-icon-section .e-btn.msg-hidden {
    display: none;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/message/getting-started-cs7" %}

Customization in Vue Message component

The Message component allows the user to customize the content display positions and appearance. This section explains the details about changing the content alignments and border styles for messages.

Content Alignment

Normally, the message content is aligned to the **left**. The Message component allows the user to align the message content in the **center** or **right** through the built-in classes `e-content-center` and `e-content-right`.

The following example demonstrates the message with different content alignments.

APP.VUE

```
<template>
  <div class="msg-custom-section">
    <div class="content-section">
      <h4>Content Alignment</h4>
      <ejs-message id="msg_content_left" content="Your license has been
activated successfully" severity="Success"></ejs-message>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      <ejs-message id="msg_content_center" content="The license will expire
today" cssClass="e-content-center" severity="Warning"></ejs-message>
      <ejs-message id="msg_content_right" content="The license key is
invalid" cssClass="e-content-right" severity="Error"></ejs-message>
    </div>
  </div>
</template>
<script setup>
import { MessageComponent as EjsMessage } from "@syncfusion/ej2-vue-
notifications";
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
  .msg-custom-section .content-section {
    margin: 0 auto;
    max-width: 400px;
    padding-top: 10px;
  }
  .msg-custom-section .e-message {
    margin: 10px 0;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/message/getting-started-cs1" %}

Rounded and Square

To customize the Message component's appearance, add the custom class to the message through the cssClass property. This custom class will be added to the root element. Based on this custom class, the user can override the message styles at the application level.

The following example shows the rounded and squared appearance of the message, which can be achieved by adding the cssClass property.

APP.VUE

```
<template>
  <div class="msg-custom-section">
    <div class="content-section">
      <h4>Rounded</h4>
      <ejs-message content="The license will expire today"
cssClass="rounded" severity="Warning"></ejs-message>
      <h4>Square</h4>
      <ejs-message content="The license key is invalid" cssClass="square"
severity="Error"></ejs-message>
    </div>
  </div>
</template>
<script setup>
import { MessageComponent as EjsMessage } from "@syncfusion/ej2-vue-
notifications";
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
  .msg-custom-section .content-section {
    margin: 0 auto;
    max-width: 400px;
    padding-top: 10px;
  }
  .msg-custom-section .e-message {
    margin: 10px 0;
  }
  .msg-custom-section .e-message.rounded {
    border-radius: 5px;
  }
  .msg-custom-section .e-message.square {
    border-radius: 1px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/message/getting-started-cs2" %}

CSS Message

The Essential JS 2 Message has predefined CSS classes that can be defined in the HTML elements, which renders the message without any script reference. This can display a simple message with content and make the code lighter.

The following DOM structure is required to display the simple message with the content.

`bash

..content..

`

The following DOM structure is required to display the simple message with the content and severity icon.

`bash

..content..

`

The following is the available list of predefined CSS classes to make the appearance of a message.

| Class | Description |

| -------- | -------- |

| e-message | Represents the message wrapper. |

Copyright © 2001 -2024 Syncfusion Inc.

| e-msg-icon | Represents the severity type icon. |

| e-msg-content | Represents the message content. |

| e-msg-close-icon | Represents the close icon. |

| e-info | Represents the information message. |

| e-success | Represents the success message. |

| e-warning | Represents the warning message. |

| e-error | Represents the error message. |

| e-content-center | Aligns the message content to the center. |

| e-content-right | Aligns the message content to the right. |

The following example shows the message which renders without any script reference.

APP.VUE

```
<template>
  <div class="msg-default">
    <div id="msg" class="e-message" role="alert">
      <span class="e-msg-icon"></span>
      <div class="e-msg-content">Please read the comments carefully</div>
    </div>
  </div>
</template>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
  .msg-default .e-message {
    margin: 100px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/message/getting-started-cs3" %}

Template in Vue Message component

The message supports templates that allows the user to customize the content with a custom structure. The content can be a string, paragraph, or any other HTML element. The template can be rendered through the `content` property or added directly to the HTML element.

In the following sample, the Message component content is customized with HTML elements and Vue Button components, which are directly added to the `<ejs-message>` element.

APP.VUE

```
<template>
  <div class=" msg-template-section">
    <div class="content-section">
      <ejs-button id="showBtn" ref="showBtn" content="Show pull request"
        cssClass="e-outline e-primary e-success msg-hidden" v-
on:click="showClick"></ejs-button>
      <ejs-message id="msg_template" ref="msgTemplate" severity="Success"
:content="'contentTemplate'" :closed="closed">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <template v-slot:contentTemplate>
          <div>
            <h1>Merged pull request</h1>
            <p>Pull request #41 merged after a successful build</p>
            <ejs-button id="commitBtn" cssClass="e-link" content="View
commit"></ejs-button>
            <ejs-button id="closeBtn" cssClass="e-link" content="Dismiss" v-
on:click="dismissClick"></ejs-button>
          </div>
        </template>
      </ejs-message>
    </div>
  </div>
</template>
<script setup>
import { MessageComponent as EjsMessage } from "@syncfusion/ej2-vue-
notifications";
import { ButtonComponent as EjsButton } from "@syncfusion/ej2-vue-buttons";
const showClick = function () {
  this.$refs.msgTemplate.ej2Instances.visible = true;
  this.$refs.showBtn.ej2Instances.element.classList.add("msg-hidden");
};
const dismissClick = function () {
  this.$refs.msgTemplate.ej2Instances.visible = false;
};
const closed = function () {
  this.$refs.showBtn.ej2Instances.element.classList.remove("msg-hidden");
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/message/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
buttons/styles/button/material.css";
.msg-template-section .content-section {
  margin: 0 auto;
  max-width: 450px;
  padding-top: 20px;
}
.msg-template-section .e-btn.msg-hidden {
  display: none;
}
.msg-template-section .e-message h1 {
  margin: 0;
  font-size: 16px;
  font-weight: 600;
  line-height: 1.25;
}
.msg-template-section .e-message .e-msg-icon {
  padding: 0 4px;
  margin-top: 3px;
}
.msg-template-section .e-message p {
  margin: 8px 0 4px;
}
.msg-template-section .e-message .e-btn {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  padding: 0;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/message/getting-started-cs9" %}

Accessibility in Vue Message component

The Message component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Message component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/not-supported.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Message component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Message component:

| Attributes | Purpose |

| --- | --- |

| role=alert | Used to convey a significant and contextual message to the user. |

| aria-label | Provides an accessible name for the close icon. |

Keyboard interaction

The Message component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Message component.

| **Press** | **To do this** |

| --- | --- |

| Tab / Shift + Tab | To focus the close icon in the message. |

| Enter / Space | Closes the focused close icon's message. |

Ensuring accessibility

The Message component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Message component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Message component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/message.html" %}

See also

- Accessibility in Syncfusion Vue components

MultiSelect

Getting Started with the Vue Multi select Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Multi select component using the Composition API / Options APILink to the Video.

To get start quickly with MultiSelect Dropdown using Vue CLI , you can check on this video:

Prerequisites

System requirements for Syncfusion Vue UI components

Copyright © 2001 -2024 Syncfusion Inc.

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

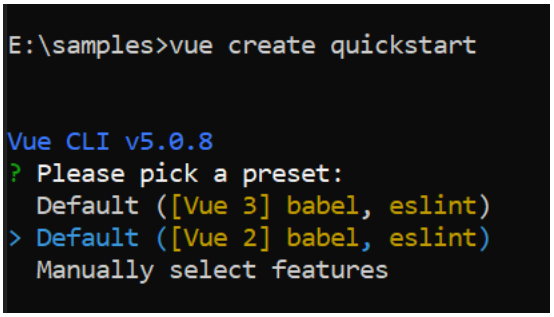
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Multi select component as an example. Install the @syncfusion/ej2-vue-dropdowns package by running the following command:

`bash

npm install @syncfusion/ej2-vue-dropdowns --save

`

or

`bash

yarn add @syncfusion/ej2-vue-dropdowns

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Multi select component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Multiselect component using Composition API or Options API:

1\. First, import and register the Multiselect component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { MultiSelectComponent as EjsMultiselect } from "@syncfusion/ej2-vue-dropdowns";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { MultiSelectComponent } from "@syncfusion/ej2-vue-dropdowns";
export default {
components: {
'ejs-multiselect': MultiSelectComponent
}
}
</script>
```

2\. In the template section, define the Multiselect component with dataSource and placeholder property

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
```

```
<div id="app">
<div id='container' style="margin:50px auto 0; width:250px;">
<br>
<ejs-multiselect id='multiselect' :dataSource='sportsData' placeholder="Find a game"></ejs-multiselect>
</div>
</div>
</template>
```

Binding data source

After initialization, populate the data using dataSource property. Here, an array of string values is passed to the MultiSelect component.

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-multiselect id='multiselect' :dataSource='sportsData'></ejs-multiselect>
</div>
</template>
<script setup>
import { MultiSelectComponent as EjsMultiselect } from "@syncfusion/ej2-vue-dropdowns";
const sportsData = ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf', 'Gymnastics', 'Hockey', 'Rugby', 'Snooker', 'Tennis'];
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-multiselect id='multiselect' :dataSource='sportsData'></ejs-multiselect>
</div>
</template>
<script>
import { MultiSelectComponent } from "@syncfusion/ej2-vue-dropdowns";
export default {
components: {
'ejs-multiselect': MultiSelectComponent
},
data: function() {
return {
sportsData: ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf', 'Gymnastics', 'Hockey', 'Rugby', 'Snooker', 'Tennis']
};
}
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sportsData'
placeholder="Find a game"></ejs-multiselect>
    </div>
  </div>
</template>
<script setup>
import { MultiSelectComponent as EjsMultiselect } from "@syncfusion/ej2-vue-
dropdowns";
const sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sportsData'
placeholder="Find a game"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import { MultiSelectComponent } from "@syncfusion/ej2-vue-dropdowns";
export default {
  components: {
    'ejs-multiselect': MultiSelectComponent
  },
  data (){
    return {
      sportsData: ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis']
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

Copyright © 2001 -2024 Syncfusion Inc.

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/multi-select/getting-started/getting-started-cs1" %}

Configure the Popup List

By default, the width of the popup list automatically adjusts according to the MultiSelect input element's width and the height of the popup list has '300px'.

You can also customize the suggestion list height and width using popupHeight and popupWidth properties respectively.

In the following sample, popup list's width and height are configured.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sportsData'
popupHeight="250px" popupWidth="250px" placeholder="Find a game"></ejs-
multiselect>
    </div>
  </div>
</template>
<script setup>
import { MultiSelectComponent as EjsMultiselect } from "@syncfusion/ej2-vue-
dropdowns";
const sportsData = ['Badminton', 'Basketball', 'Cricket', 'Football',
'Golf', 'Gymnastics', 'Hockey', 'Rugby','Snooker', 'Tennis'];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sportsData'
popupHeight="250px" popupWidth="250px" placeholder="Find a game"></ejs-
multiselect>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
  </div>
</template>
<script>
import { MultiSelectComponent } from "@syncfusion/ej2-vue-dropdowns";
export default {
  components: {
    'ejs-multiselect': MultiSelectComponent
  },
  data (){
    return {
      sportsData: ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf',
'Gymnastics', 'Hockey', 'Rugby','Snooker', 'Tennis']
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/getting-started/popup-cs1" %}

See Also

- How to bind the data

Getting Started with the Vue MultiSelect Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue MultiSelect component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

Copyright © 2001 -2024 Syncfusion Inc.

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

Copyright © 2001 -2024 Syncfusion Inc.

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue MultiSelect component as an example. To use the Vue MultiSelect component in the project, the @syncfusion/ej2-vue-dropdowns package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-dropdowns --save

`

or

`bash

yarn add @syncfusion/ej2-vue-dropdowns

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the MultiSelect component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Copyright © 2001 -2024 Syncfusion Inc.

Add Syncfusion Vue component

Follow the below steps to add the Vue MultiSelect component using Composition API or Options API:

1.First, import and register the MultiSelect component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { MultiSelectComponent as EjsMultiselect } from "@syncfusion/ej2-vue-dropdowns";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { MultiSelectComponent } from "@syncfusion/ej2-vue-dropdowns";
//Component registeration
export default {
name: "App",
components: {
'ejs-multiselect' : MultiSelectComponent,
}
}
</script>
```

2.In the template section, define the MultiSelect component with the dataSource property and column definitions.

~/SRC/APP.VUE

```
<template>
<div class="control_wrapper">
<ejs-multiselect id='multiselect' :dataSource='sportsData'></ejs-combobox>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-multiselect id='multiselect' :dataSource='data[0].sportsData'></ejs-multiselect>
</div>
</template>
<script setup>
import { MultiSelectComponent as EjsMultiselect } from "@syncfusion/ej2-vue-dropdowns";
const data = [{ sportsData: ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf', 'Gymnastics', 'Hockey', 'Rugby', 'Snooker', 'Tennis']}]
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-multiselect id='multiselect' :dataSource='sportsData'></ejs-
multiselect>
</div>
</template>
<script>
import { MultiSelectComponent } from "@syncfusion/ej2-vue-dropdowns";
//Component registeration
export default {
name: 'App',
components: {
"ejs-multiselect": MultiSelectComponent
},
data () {
return {
sportsData: ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf',
'Gymnastics', 'Hockey', 'Rugby', 'Snooker', 'Tennis']
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

Configure the Popup List

By default, the width of the popup list automatically adjusts according to the MultiSelect input element's width and the height of the popup list has '300px'.

Copyright © 2001 -2024 Syncfusion Inc.

You can also customize the suggestion list height and width using popupHeight and popupWidth properties respectively.

In the following sample, popup list's width and height are configured.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div id='container' style="margin:50px auto 0; width:250px;">
<br>
<ejs-multiselect id='multiselect' :dataSource='data[0].sportsData'
popupHeight="250px" popupWidth="250px" placeholder="Find a game"></ejs-
multiselect>
</div>
</div>
</template>
<script setup>
import { MultiSelectComponent as EjsMultiselect } from "@syncfusion/ej2-vue-
dropdowns";
const data = [{ sportsData: ['Badminton', 'Basketball', 'Cricket',
'Football', 'Golf', 'Gymnastics', 'Hockey', 'Rugby', 'Snooker', 'Tennis'] }]
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

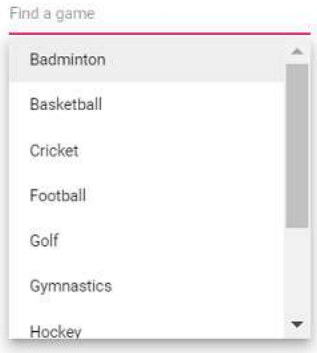
OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div id='container' style="margin:50px auto 0; width:250px;">
<br>
<ejs-multiselect id='multiselect' :dataSource='sportsData'
popupHeight="250px" popupWidth="250px" placeholder="Find a game"></ejs-
multiselect>
</div>
</div>
</template>
<script>
import { MultiSelectComponent } from "@syncfusion/ej2-vue-dropdowns";
//Component registeration
export default {
name: 'App',
components: {
"ejs-multiselect": MultiSelectComponent
},
data (){
return {
sportsData: ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf',
'Gymnastics', 'Hockey', 'Rugby', 'Snooker', 'Tennis']
}
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

The output will appear as follows:



See Also

- How to bind the data

Data binding in Vue Multi select component

The MultiSelect loads the data either from local data sources or remote data services using the dataSource property. It supports the data type of `array` or `DataManager`.

The MultiSelect also supports different kinds of data services such as OData, OData V4, and Web API, and data formats such as XML, JSON, and JSONP with the help of `DataManager` adaptors.

| Fields | Type | Description |

|------|------|-------------|

| text | `string` | Specifies the display text of each list item. |

| value | `number or string` | Specifies the hidden data value mapped to each list item that should contain a unique value. |

| groupBy | `string` | Specifies the category under which the list item has to be grouped. |

| iconCss | `string` | Specifies the icon class of each list item. |

When binding complex data to the MultiSelect, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in two ways as described below.

1. Array of string

The MultiSelect has support to load array of primitive data such as strings and numbers. Here, both value and text field act the same.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sportsData'
placeholder="Select a game"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      sportsData: ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis']
    }
  }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/data-binding/simple-cs1" %}

2. Array of object

The MultiSelect can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the fields property.

In the following example, id column and sports column from complex data have been mapped to the value field and text field, respectively.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sportsData'
:fields='fields' placeholder="Select a game"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      sportsData: [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' }
      ],
      fields : { text: 'Game', value: 'Id' }
    }
  }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/data-binding/json-cs1" %}

3. Array of complex object

The MultiSelect can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the fields property.

In the following example, Code.Id column and Country.Name column from complex data have been mapped to the value field and text field, respectively.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='countriesData'
:fields='fields' placeholder="Select a country"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      countriesData : [
        { Country: { Name: 'Australia' }, Code: { Id: 'AU' }},
        { Country: { Name: 'Bermuda' },Code: { Id: 'BM' }},
        { Country:{ Name: 'Canada'}, Code:{ Id: 'CA'} },
        { Country:{Name: 'Cameroon'}, Code:{ Id: 'CM'} },
```

```
        { Country:{Name: 'Denmark'}, Code:{ Id: 'DK' }},
        { Country:{Name: 'France'}, Code: { Id:'FR'} }
      ],
      fields : { text: 'Country.Name', value: 'Code.Id' }
    }
  }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/data-binding/complex-cs1" %}

Binding remote data

The MultiSelect supports retrieval of data from remote data services with the help of DataManager component. The Query property is used to fetch data from the database and bind it to the MultiSelect.

The following sample displays the first 6 contacts from “Customers” table of the Northwind Data Service.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' sortOrder='Ascending'
:dataSource='customerData' :query='query' :fields='fields'
placeholder="Select a customer"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
export default {
  data (){
    return {
      customerData : new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
      }),
      query : new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6),
      fields : { text: 'ContactName', value: 'CustomerID' }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/data-binding/remote-cs1" %}

See Also

- How to load data using template
- How to group the data using header
- How to filter the bound dataLink to the Video

Templates in Vue Multi select component

The MultiSelect has been provided with several options to customize each list item, group title, selected value, header, and footer elements.

To customize MultiSelect dropdown items Using templates, you can check on this video:

Item template

The content of each list item within the MultiSelect can be customized with the help of itemTemplate property.

In the following sample, each list item is split into two columns to display relevant data's.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' sortOrder='Ascending'
:itemTemplate='itemTemplate' :dataSource='employeeData' :query='query'
:fields='fields' placeholder="Select an employee"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
var itemVue = Vue.component("itemTemplate", {
  template: `<span><span class='name'>{{data.FirstName}}</span><span class
='city'>{{data.City}}</span></span>`,
  data() {
    return {
      data: {}
    };
```

```
  }
});
export default {
  data (){
    return {
      employeeData : new DataManager({
            url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
            adaptor: new ODataV4Adaptor,
            crossDomain: true
      }),
      query : new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6),
      fields : { text: 'FirstName', value: 'EmployeeID' },
      itemTemplate : function(e) {
        return {
          template: itemVue
        }
      }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.city {
    right: 15px;
    position: absolute;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/template/item-cs1" %}

Value template

The currently selected value that is displayed by default on the MultiSelect input element can be customized using the valueTemplate property.

In the following sample, the selected value is displayed as a combined text of both FirstName and City in the MultiSelect input, which is separated by a hyphen.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' placeholder='Select an employee'
sortOrder='Ascending' :valueTemplate="valueTemplate"
:itemTemplate="itemTemplate" :dataSource='dataSource' :query='query'
:fields='fields'></ejs-multiselect>
    </div>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
import { DataManager,Query,ODataV4Adaptor } from "@syncfusion/ej2-data";
var itemVue = Vue.component("itemTemplate", {
  template: `<span><span class='name'>{{data.FirstName}}</span><span class
='city'>{{data.City}}</span></span>`,
  data() {
    return {
      data: {}
    };
  }
});
var valueVue = Vue.component("valueTemplate", {
  template: `<span>{{data.FirstName}} - {{data.City}}</span>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data (){
      return {
        valueTemplate : function(e) {
          return {
            template: valueVue
          }
        },
        itemTemplate : function(e) {
          return {
            template: itemVue
          }
        },
        query :  new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6),
        dataSource : new DataManager({
            url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
            adaptor: new ODataV4Adaptor,
            crossDomain: true
        }),
        fields: { text: 'FirstName', value: 'EmployeeID' }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.city {
    right: 15px;
    position: absolute;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/template/value-cs1" %}

Group template

The group header title under which appropriate sub-items are categorized can also be customize with the help of groupTemplate property. This template is common for both inline and floating group header template.

In the following sample, employees are grouped according to their city.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' sortOrder='Ascending'
:groupTemplate='groupTemplate' :dataSource='employeeData' :query='query'
:fields='fields' placeholder="Select an employee"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
import { Query, DataManager, ODataV4Adaptor, Predicate } from
'@syncfusion/ej2-data';
var itemVue = Vue.component("groupTemplate", {
  template: `<strong>{{data.City}}</strong>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data (){
    return {
      employeeData : new DataManager({
            url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
            adaptor: new ODataV4Adaptor,
            crossDomain: true
      }),
      query : new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6).where(new Predicate('City', 'equal',
'london').or('City', 'equal', 'seattle')),
      fields : { text: 'FirstName', value: 'EmployeeID', groupBy: 'City' },
      groupTemplate : function(e) {
        return {
          template: itemVue
        }
      }
    }
  }
```

```
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/template/group-cs1" %}

Header template

The header element is shown statically at the top of the popup list items within the MultiSelect, and any custom element can be placed as a header element using the headerTemplate property.

In the following sample, the list items and its headers are designed and displayed as two columns similar to multiple columns of the grid.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' sortOrder='Ascending'
:headerTemplate='headerTemplate' :itemTemplate='itemTemplate'
:dataSource='employeeData' :query='query' :fields='fields'
placeholder="Select an employee"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
var headerVue = Vue.component("headerTemplate", {
  template: `<span class='head'><span class='name'>Name</span><span
class='city'>City</span></span>`,
  data() {
    return {
      data: {}
    };
  }
});
var itemVue = Vue.component("itemTemplate", {
  template: `<span class='item' ><span
class='name'>{{data.FirstName}}</span><span
class='city'>{{data.City}}</span></span>`,
  data() {
    return {
      data: {}
    };
  }
});
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data (){
    return {
      employeeData : new DataManager({
            url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
            adaptor: new ODataV4Adaptor,
            crossDomain: true
      }),
      query : new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6),
      fields : { text: 'FirstName', value: 'EmployeeID' },
      headerTemplate : function(e) {
        return {
          template: headerVue
        }
      },
      itemTemplate : function(e) {
        return {
          template: itemVue
        }
      }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.head,
.item {
    display: table;
    width: 100%;
    margin: auto;
}
.head {
    height: 40px;
    font-size: 15px;
    font-weight: 600;
}
.name,
.city {
    display: table-cell;
    vertical-align: middle;
    width: 50%;
}
.head .name {
    text-indent: 16px;
 }
 .head .city {
    text-indent: 10px;
 }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/multi-select/template/header-cs1" %}

Footer template

The MultiSelect has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the footerTemplate property.

In the following sample, footer element displays the total number of list items present in the MultiSelect.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sportsData'
:footerTemplate='footerTemplate' :open='onPopupOpen'
:select='updateDataCount' placeholder="Select a game"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { isNullOrUndefined } from "@syncfusion/ej2-base";
Vue.use(MultiSelectPlugin);
var footerVue = Vue.component("footerTemplate", {
  template: `<span class='foot'></span>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data (){
    var data = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
    return {
      sportsData: data,
      footerTemplate : function(e) {
        return {
          template: footerVue
        }
      }
    }
  },
  methods: {
    updateDataCount: function() {
      var remainingDatasCount = document.querySelectorAll('.e-list-
item').length - document.querySelectorAll('.e-hide-listitem').length;
      if (remainingDatasCount !== 0) {
        document.querySelector('.foot').innerText = "Total list item: " +
(remainingDatasCount -1);
      }
    },
    onPopupOpen: function(e) {
```

```
        e.popup.element.querySelector('.foot').innerText  = "Total list
item: " + (5- document.querySelectorAll('.e-chips-collection .e-
chips').length);
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.foot {
    text-indent: 1.2em;
    display: block;
    font-size: 15px;
    line-height: 40px;
    border-top: 1px solid #e0e0e0;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/template/footer-cs1" %}

No records template

The MultiSelect is provided with support to custom design the popup list content when no data is found and no matches found on search with the help of noRecordsTemplate property.

In the following sample, popup list content displays the notification of no data available.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='data'
:noRecordsTemplate='noRecordsTemplate' placeholder="Select an item"></ejs-
multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
var footerVue = Vue.component("footerTemplate", {
  template: `<span class='norecord'> NO DATA AVAILABLE</span>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data (){
```

```
    return {
      data: [],
      noRecordsTemplate : function(e) {
        return {
          template: footerVue
        }
      }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/template/no-records-cs1" %}

Action failure template

There is also an option to custom design the popup list content when the data fetch request fails at the remote server. This can be achieved using the actionFailureTemplate property.

In the following sample, when the data fetch request fails, the MultiSelect displays the notification.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect'
:actionFailureTemplate='actionFailureTemplate' :dataSource='customerData'
:query='query' :fields='fields' placeholder="Select a customer"></ejs-
multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
var footerVue = Vue.component("footerTemplate", {
  template: `<span class='action-failure'> Data fetch get fails</span>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data (){
    return {
```

```
      customerData : new DataManager({
            url: 'https://services.odata.org/V4/Northwind/Northwind.svcs/',
            adaptor: new ODataV4Adaptor,
            crossDomain: true
      }),
      query : new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6),
      fields : { text: 'ContactName', value: 'CustomerID' },
      actionFailureTemplate : function (e) {
        return {
          template: footerVue
        }
      }
      }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/template/action-cs1" %}

See Also

- How to bind the data
- How to group the data using header
- How to customize the options in MultiSelect

Grouping in Vue Multi select component

The MultiSelect supports wrapping nested elements into a group based on different categories. The category of each list item can be mapped through the groupByLink to the Video field in the data table. The group header is displayed both as inline and fixed headers. The fixed group header contentis updated dynamically on scrolling the popup list with its category value.

To group the Vue MultiSelect dropdown items, you can check on this video:

In the following sample, vegetables are grouped according on its category using groupBy field.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='vegetableData'
:fields='fields' popupHeight="200px" placeholder="Select a vegetable"></ejs-
multiselect>
    </div>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      vegetableData: [
        { vegetable: 'Cabbage', category: 'Leafy and Salad', id: 'item1' },
        { vegetable: 'Spinach', category: 'Leafy and Salad', id: 'item2' },
        { vegetable: 'Wheat grass', category: 'Leafy and Salad', id: 'item3'
},
        { vegetable: 'Yarrow', category: 'Leafy and Salad', id: 'item4' },
        { vegetable: 'Pumpkins', category: 'Leafy and Salad', id: 'item5' },
        { vegetable: 'Chickpea', category: 'Beans', id: 'item6' },
        { vegetable: 'Green bean', category: 'Beans', id: 'item7' },
        { vegetable: 'Horse gram', category: 'Beans', id: 'item8' },
        { vegetable: 'Garlic', category: 'Bulb and Stem', id: 'item9' },
        { vegetable: 'Nopal', category: 'Bulb and Stem', id: 'item10' },
        { vegetable: 'Onion', category: 'Bulb and Stem', id: 'item11' }
    ],
    fields : { groupBy: 'category', text: 'vegetable', value: 'id' }
    }
  }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/grouping-cs1" %}

Customization

The grouping header is also provided with customization option. This allows custom designing using the groupTemplate property for both inline and fixed headers.

Grouping with CheckBox

Previously, there is no checkbox for group headers. Now, this feature allow to render checkbox in group header to select the group items in single selection. You can enable this feature by setting enableGroupCheckBox property value as **true** and **mode** property as **CheckBox**.

Inject the CheckBoxSelection module in the MultiSelect to use the checkbox.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <ejs-multiselect id='multiselect' :dataSource='vegetableData'
:fields='fields' popupHeight="200px" mode="CheckBox"
enableGroupCheckBox="true" allowFiltering="true"
filterBarPlaceholder="Search Vegetables" showSelectAll="true"
placeholder="Select vegetables"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
MultiSelect.Inject(CheckBoxSelection);
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      vegetableData: [
        { vegetable: 'Cabbage', category: 'Leafy and Salad', id: 'item1' },
        { vegetable: 'Spinach', category: 'Leafy and Salad', id: 'item2' },
        { vegetable: 'Wheat grass', category: 'Leafy and Salad', id: 'item3'
},
        { vegetable: 'Yarrow', category: 'Leafy and Salad', id: 'item4' },
        { vegetable: 'Pumpkins', category: 'Leafy and Salad', id: 'item5' },
        { vegetable: 'Chickpea', category: 'Beans', id: 'item6' },
        { vegetable: 'Green bean', category: 'Beans', id: 'item7' },
        { vegetable: 'Horse gram', category: 'Beans', id: 'item8' },
        { vegetable: 'Garlic', category: 'Bulb and Stem', id: 'item9' },
        { vegetable: 'Nopal', category: 'Bulb and Stem', id: 'item10' },
        { vegetable: 'Onion', category: 'Bulb and Stem', id: 'item11' }
    ],
    fields : { groupBy: 'category', text: 'vegetable', value: 'id' }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/grouping-with-checkbox-cs1" %}

See Also

- Group Template support to MultiSelect.

Filtering in Vue Multi select component

The MultiSelect has built-in support to filter data items when allowFiltering is enabled. The filter operation starts as soon as you start typing characters in the MultiSelect input.

Copyright © 2001 -2024 Syncfusion Inc.

To display filtered items in the popup, filter the required data and return it to the MultiSelect via Link to the Video method by using the filtering event.

To filter the Vue MultiSelect dropdown items, you can check on this video:

The following sample illustrates how to query the data source and pass the data to the MultiSelect through the updateData method in filtering event.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' popupHeight="250px"
:allowFiltering='allowFiltering' :filtering='filtering'
:dataSource='searchData' :fields='fields' placeholder="Select a
country"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
import { Query } from '@syncfusion/ej2-data';
export default {
  data (){
    return {
        searchData : [
            { Index: "s1", Country: "California" }, { Index: "s2", Country:
"Florida" },
            { Index: "s3", Country: "Alaska" }, { Index: "s4", Country:
"Georgia" }
        ],
        fields : { text: "Country", value: "Index" },
        allowFiltering : true
        }
  },
   methods: {
        filtering: function(args) {
           var searchData = [
                { Index: "s1", Country: "California" }, { Index: "s2",
Country: "Florida" },
                { Index: "s3", Country: "Alaska" }, { Index: "s4", Country:
"Georgia" }
            ];
           var query = new Query();
            //frame the query based on search string with filter type.
           query = (args.text != "") ? query.where("Country", "startswith",
args.text, true) : query;
            //pass the filter data source, filter query to updateData
method.
            args.updateData(searchData, query);
        }
    }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/filtering/default-cs1" %}

Limit the minimum filter character

When filtering the list items, you can set the limit for character count to raise remote request and fetch filtered data on the MultiSelect. This can be done by manual validation within the filter event handler.

In the following example, the remote request does not fetch the search data until the search key contains three characters.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' sortOrder='Ascending'
popupHeight="250px" :allowFiltering='allowFiltering' :filtering='filtering'
:dataSource='searchData' :fields='fields' placeholder="Select a
customer"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
export default {
  data (){
    return {
        searchData : new DataManager({
            url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
            adaptor: new ODataV4Adaptor,
            crossDomain: true
        }),
        fields : { text: 'ContactName', value: 'CustomerID' },
        query : new Query().select(['ContactName', 'CustomerID']).take(6),
        allowFiltering : true
        }
  },
   methods: {
        filtering: function(e) {
           var searchData = new DataManager({
                url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
                adaptor: new ODataV4Adaptor,
```

```
                crossDomain: true
            });
           // load overall data when search key empty.
            if (e.text == '') e.updateData(searchData);
            else {
                // restrict the remote request until search key contains 3
characters.
                if (e.text.length < 3) { return; }
                var query = new Query().select(['ContactName',
'CustomerID']);
                query = (e.text !== '') ? query.where('ContactName',
'startswith', e.text, true) : query;
                e.updateData(searchData, query);
            }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/filtering/min-char-cs1" %}

Change the filter type

While filtering, you can change the filter type to `contains`, `startsWith`, or `endsWith` for string type within the filter event handler.

In the following examples, data filtering is done with `endsWith` type.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' sortOrder='Ascending'
popupHeight="250px" :allowFiltering='allowFiltering' :filtering='filtering'
:dataSource='searchData' :query='query' :fields='fields' placeholder="Select
a customer"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
export default {
  data (){
    return {
             searchData : new DataManager({
```

```
                url:
'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Suppliers',
                crossDomain: true
            }),
            fields : { text: "ContactName", key: "SupplierID" },
            query : new Query().select(["SupplierID",
"ContactName"]).take(6),
            allowFiltering : true
        }
  },
   methods: {
        filtering: function(e) {
           var searchData = new DataManager({
                url:
'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Suppliers',
                crossDomain: true
            });
             if (e.text == '') e.updateData(searchData);
            else {
                var query = new Query().select(["SupplierID",
"ContactName"]);
                query = (e.text !== '') ? query.where('ContactName',
'endswith', e.text, true) : query;
                e.updateData(searchData, query);
            }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/filtering/type-cs1" %}

Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by passing the fourth optional parameter of the where clause.

The following example shows how to perform case-sensitive filter.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' sortOrder='Ascending'
:allowFiltering='allowFiltering' :filtering='filtering'
:dataSource='sportsData' :fields='fields' placeholder="Select a game"></ejs-
multiselect>
    </div>
```

```
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
import { Query } from '@syncfusion/ej2-data';
export default {
  data (){
    return {
             sportsData : [
                 { id: 'game1', sports: 'Badminton' },
                 { id: 'game2', sports: 'Tennis' },
                 { id: 'game3', sports: 'Football' }
             ],
             fields : { text: 'sports', value: 'id' }
             allowFiltering : true,
        }
  },
   methods: {
        filtering: function(e) {
           var sportsData = [
                 { id: 'game1', sports: 'Badminton' },
                 { id: 'game2', sports: 'Tennis' },
                 { id: 'game3', sports: 'Football' }
            ];
             if (e.text == '') e.updateData(sportsData);
             else {
                 var query = new Query().select(["sports", "id"]);
                 //enable the case sensitive filtering by passing false to
4th parameter.
                 query = (e.text !== '') ? query.where('sports',
'startsWith', e.text, false) : query;
                 e.updateData(sportsData, query);
             }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/filtering/case-cs1" %}

Diacritics Filtering

MultiSelect supports diacritics filtering which will ignore the diacritics and makes it easier to filter the results in international characters lists when the ignoreAccent is enabled.

In the following sample,data with diacritics are bound as dataSource for MultiSelect.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:15px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='diacriticsData'
:ignoreAccent="ignoreAccent" :allowFiltering='allowFiltering'
placeholder="e.g: aero"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      diacriticsData : [
            'Aeróbics',
            'Aeróbics en Agua',
            'Aerografía',
            'Aeromodelaje',
            'Águilas',
            'Ajedrez',
            'Ala Delta',
            'Álbumes de Música',
            'Alusivos',
            'Análisis de Escritura a Mano'
        ],
        ignoreAccent : true,
        allowFiltering : true
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/filtering/diacritics-cs1" %}

See Also

- How to bind the data
- How to group the data using header
- How to add custom value to the MultiSelect

Copyright © 2001 -2024 Syncfusion Inc.

Custom value in Vue Multi select component

The MultiSelect allows user to add a new non-present option to the component value when allowCustomValue is enabled. while selecting the new custom value customValueSelection event will be triggered.

The following sample demonstrates configuration of custom value support with the MultiSelect component.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sportsData'
:fields='fields' :allowCustomValue='allowCustomValue' placeholder="Select a
game"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      sportsData: [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' }
      ],
      fields : { text: 'Game', value: 'Id' },
      allowCustomValue : true
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/custom-value-cs1" %}

Checkbox in Vue Multi select component

The MultiSelect has built-in support to select multiple values through checkbox, when mode property set as `CheckBox`.

To use checkbox, inject the `CheckBoxSelection` module in the MultiSelect.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:15px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sportsData'
placeholder="Find a game" mode="CheckBox" :fields='fields'></ejs-
multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
MultiSelect.Inject(CheckBoxSelection);
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      sportsData: [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' },
        { Id: 'game4', Game: 'Golf' },
        { Id: 'game5', Game: 'Cricket' },
        { Id: 'game6', Game: 'Handball' },
        { Id: 'game7', Game: 'Karate' },
        { Id: 'game8', Game: 'Fencing' },
        { Id: 'game9', Game: 'Boxing' }
      ],
      fields : { text: 'Game', value: 'Id' }
    }
  }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/checkbox/default-cs1" %}

Select All

The MultiSelect component has in-built support to select the all list items using Select All options in the header.

When the showSelectAll property is set to true, by default Select All text will show. You can customize the name attribute of the Select All option by using selectAllText.

For the unSelect All option, by default unSelect All text will show. You can customize the name attribute of the unSelect All option by using

unSelectAllText.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:15px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sportsData'
placeholder="Select a game" mode="CheckBox" :fields='fields'
:showSelectAll='showSelectAll' selectAllText="Select All"
unSelectAllText="unSelect All"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
MultiSelect.Inject(CheckBoxSelection);
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      sportsData: [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' },
        { Id: 'game4', Game: 'Golf' },
        { Id: 'game5', Game: 'Cricket' },
        { Id: 'game6', Game: 'Handball' },
        { Id: 'game7', Game: 'Karate' },
        { Id: 'game8', Game: 'Fencing' },
        { Id: 'game9', Game: 'Boxing' }
      ],
      fields : { text: 'Game', value: 'Id' },
      showSelectAll : true
    }
  }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/checkbox/selectall-cs1" %}

Selection Limit

Defines the limit of the selected items using maximumSelectionLength.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:15px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sportsData'
placeholder="Select a game" mode="CheckBox" :fields='fields'
:maximumSelectionLength='maximumSelectionLength'></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
MultiSelect.Inject(CheckBoxSelection);
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      sportsData: [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' },
        { Id: 'game4', Game: 'Golf' },
        { Id: 'game5', Game: 'Cricket' },
        { Id: 'game6', Game: 'Handball' },
        { Id: 'game7', Game: 'Karate' },
        { Id: 'game8', Game: 'Fencing' },
        { Id: 'game9', Game: 'Boxing' }
      ],
      fields : { text: 'Game', value: 'Id' },
      maximumSelectionLength : 3
    }
  }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/checkbox/selectionlimit-cs1" %}

Selection Reordering

Using enableSelectionOrder to Reorder the selected items in popup visibility state.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:15px auto 0; width:250px;">
```

```
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sportsData'
placeholder="Select a game" mode="CheckBox" :fields='fields'
:enableSelectionOrder='enableSelectionOrder'></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { MultiSelect, CheckBoxSelection } from '@syncfusion/ej2-dropdowns';
MultiSelect.Inject(CheckBoxSelection);
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      sportsData: [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' },
        { Id: 'game4', Game: 'Golf' },
        { Id: 'game5', Game: 'Cricket' },
        { Id: 'game6', Game: 'Handball' },
        { Id: 'game7', Game: 'Karate' },
        { Id: 'game8', Game: 'Fencing' },
        { Id: 'game9', Game: 'Boxing' }
      ],
      fields : { text: 'Game', value: 'Id' },
      enableSelectionOrder : false
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/checkbox/selectionreorder-cs1" %}

See Also

- How to bind the data
- How to filter the bound data
- How to add custom value to the MultiSelect
- How to render grouping with checkbox.

Chip customization in Vue Multi select component

The MultiSelect allows the user to customize the selected chip element through the tagging event. In that event, you can set the custom classes to chip element via that event argument of setClass method.

Copyright © 2001 -2024 Syncfusion Inc.

The following sample demonstrates chip-customization with the MultiSelect component.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :value='colorValues'
:tagging='tagging' :dataSource='colorsData' mode="Box" placeholder="Favorite
Colors" :fields='fields'></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { MultiSelect, TaggingEventArgs } from '@syncfusion/ej2-dropdowns';
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      colorsData: [
        { Color: 'Chocolate', Code: '#75523C' },
        { Color: 'CadetBlue', Code: '#3B8289' },
        { Color: 'DarkOrange', Code: '#FF843D' },
        { Color: 'DarkRed', Code: '#CA3832' },
        { Color: 'Fuchsia', Code: '#D44FA3' },
        { Color: 'HotPink', Code: '#F23F82' },
        { Color: 'Indigo', Code: '#2F5D81' },
        { Color: 'LimeGreen', Code: '#4CD242' },
        { Color: 'OrangeRed', Code: '#FE2A00' },
        { Color: 'Tomato', Code: '#FF745C' }
      ],
      colorValues : ['#75523C', '#4CD242', '#FF745C'],
      fields : { text: 'Color', value: 'Code' }
    }
  },
  methods: {
        tagging: function(e) {
           var msObject =
document.getElementById("multiselect").ej2_instances[0];
          e.setClass(e.itemData[msObject.fields.text].toLowerCase());
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    .e-multi-select-wrapper .e-chips {
        opacity: 0.9;
    }
    .e-multi-select-wrapper .e-chips:hover {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        opacity: 1;
    }
    .e-multi-select-wrapper .e-chips .e-chips-close.e-icon::before,
    .e-multi-select-wrapper .e-chips .e-chipcontent,
    .e-multi-select-wrapper .e-chips .e-chipcontent:hover {
        color: #ffffff;
    }
    .e-chips.chocolate,
    .e-chips.chocolate:hover {
        background-color: #75523C;
    }
    .e-chips.darkorange,
    .e-chips.darkorange:hover {
        background-color: #FF843D;
    }
    .e-chips.darkred,
    .e-chips.darkred:hover {
        background-color: #CA3832;
    }
    .e-chips.fuchsia,
    .e-chips.fuchsia:hover {
        background-color: #D44FA3;
    }
    .e-chips.cadetblue,
    .e-chips.cadetblue:hover {
        background-color: #3B8289;
    }
    .e-chips.hotpink,
    .e-chips.hotpink:hover {
        background-color: #F23F82;
    }
    .e-chips.indigo,
    .e-chips.indigo:hover {
        background-color: #2F5D81;
    }
    .e-chips.limegreen,
    .e-chips.limegreen:hover {
        background-color: #4CD242;
    }
    .e-chips.orangered,
    .e-chips.orangered:hover {
        background-color: #FE2A00;
    }
    .e-chips.tomato,
    .e-chips.tomato:hover {
        background-color: #FF745C;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/chip-customization-cs1" %}

Localization in Vue Multi select component

The Localization library allows you to localize static text content of the noRecordsTemplate and actionFailureTemplate properties according to the culture currently assigned to the MultiSelect.

| Locale key | en-US (default) |

|-----------|-----------------|

| noRecordsTemplate | No records found |

| actionFailureTemplate | The request failed |

Loading translations

To load translation object to your application, use load function of the **L10n** class.

In the following sample, French culture is set to the MultiSelect and no data is loaded. Hence, the noRecordsTemplate property displays its text in French culture initially, and if the sample is run offline, the actionFailureTemplate property displays its text appropriately.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='customerData'
:query='query' :fields='fields' locale="fr-BE" placeholder="Sélectionnez un
client"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { L10n } from '@syncfusion/ej2-base';
Vue.use(MultiSelectPlugin);
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
L10n.load({
            'fr-BE': {
                'multi-select': {
                    'noRecordsTemplate': "Aucun enregistrement trouvé",
                    'actionFailureTemplate': "Modèle d'échec d'action"
                }
            }
        });
export default {
  data (){
    return {
      customerData : new DataManager({
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
      }),
      query : new Query().select(['ContactName', 'CustomerID']).take(0),
      fields : { text: 'ContactName', value: 'CustomerID' }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/locale-cs1" %}

See Also

- Accessibility
- How to bind the data to the combobox

Style in Vue Multi select component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the background color of wrapper element

Use the following CSS to customize the background color of wrapper element.

`

.e-multiselect.e-input-group .e-multi-select-wrapper {

background-color: red;

}

`

Customizing the appearance of the delimiter wrapper element

Use the following CSS to customize the appearance of delimiter wrapper element.

`

.e-multiselect .e-delim-values {

-webkit-text-fill-color: blue;

font-size: 16px;

font-family: cursive;

}

`

Customizing the appearance of chips

Use the following CSS to customize the appearance of selected chips.

`

.e-multiselect .e-multi-select-wrapper .e-chips .e-chipcontent {

font-family: cursive;

font-size: 20px;

-webkit-text-fill-color: blue;

Copyright © 2001 -2024 Syncfusion Inc.

```
}
.e-multi-select-wrapper .e-chips {
background-color: aqua;
height: 26px;
}
```

Customizing the dropdown icon’s color

Use the following CSS to customize the dropdown icon’s color.

```
.e-multiselect.e-input-group .e-input-group-icon, .e-multiselect.e-input-group.e-control-wrapper .e-input-group-icon:hover {
color: red;
font-size: 14px;
}
```

Customizing the focus color

Use the following CSS to customize the focusing color of input element.

```
.e-multiselect.e-input-group.e-control-wrapper.e-input-focus::before, .e-multiselect.e-input-group.e-control-wrapper.e-input-focus::after {
background: #c000ff;
}
```

Customizing the disabled component’s text color

Use the following CSS to customize the text color when the component is disabled.

```
.e-multiselect.e-disabled .e-multi-select-wrapper .e-delim-values {
-webkit-text-fill-color: red;
}
```

Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

```
.e-multiselect input.e-dropdownbase::placeholder {
color: red;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
```

Customizing the placeholder to add mandatory indicator(*)

Use the following CSS to add the mandatory indicator * to the float label element.

```
.e-input-group.e-control-wrapper.e-float-input .e-float-text::after {
content: "*";
color: red;
}
```

Customizing the float label element's focusing color

Use the following CSS to customize the focusing color of float label element.

```
.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::after,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::after {
background-color: #2319b8;
}
.e-multiselect.e-input-group.e-control-wrapper.e-float-input.e-input-focus .e-float-text.e-label-top, .e-float-input.e-control-wrapper:not(.e-error).e-input-focus input ~ label.e-float-text {
color: #2319b8;
}
```

Customizing the outline theme's focus color

Use the following CSS to customize the focusing color of outline theme.

```
.e-outline.e-input-group.e-input-focus:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus.e-control-wrapper:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled),.e-outline.e-input-group.e-control-wrapper.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled) {
border-color: #b1bd15;
box-shadow: inset 1px 1px #b1bd15, inset -1px 0 #b1bd15, inset 0 -1px #b1bd15;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

Customizing the background color of focus, hover, and active item's

Use the following CSS to customize the background color of focus, hover and active item's.

```
.e-dropdownbase .e-list-item.e-item-focus, .e-dropdownbase .e-list-item.e-active, .e-dropdownbase .e-list-item.e-active.e-hover, .e-dropdownbase .e-list-item.e-hover {
background-color: #1f9c99;
color: #2319b8;
}
```

Customizing the appearance of pop-up element

Use the following CSS to customize the appearance of popup element.

```
.e-dropdownbase .e-list-item, .e-dropdownbase .e-list-item.e-item-focus {
background-color: #29c2b8;
color: #207cd9;
font-family: emoji;
min-height: 29px
}
```

Customizing the color of the checkbox

Use the following CSS to customize the color of checkbox.

```
.e-popup .e-checkbox-wrapper .e-frame.e-check, .e-popup .e-checkbox-wrapper:hover .e-frame.e-check {
background-color: green;
color: white;
}
```

Accessibility in Vue Multi select component

The MultiSelect component has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties along with keyboard support. This component is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The MultiSelect component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

Copyright © 2001 -2024 Syncfusion Inc.

The accessibility compliance for the MultiSelect component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | |

| Section 508 Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| Accessibility Checker Validation | |

| Axe-core Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

 - All features of the component meet the requirement.

 - Some features of the component do not meet the requirement.

 - The component does not meet the requirement.

WAI-ARIA attributes

The MultiSelect component uses the `Listbox` role, and each list item has an `option` role. The following `ARIA attributes` denote the MultiSelect state.

| **Properties** | **Functionalities** |

Copyright © 2001 -2024 Syncfusion Inc.

| --- | --- |

| aria-haspopup | Indicates whether the MultiSelect input element has a popup list or not. |

| aria-expanded | Indicates whether the popup list has expanded or not. |

| aria-selected | Indicates the selected option. |

| aria-readonly | Indicates the readonly state of the MultiSelect element. |

| aria-disabled | Indicates whether the MultiSelect component is in a disabled state or not. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

Keyboard interaction

You can use the following key shortcuts to access the MultiSelect without interruptions.

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Arrow Down | Set focus at the first item in the MultiSelect when no item selected. Otherwise, moves focus next to the currently selected item. |

| Arrow Up | Moves focus previous to the currently selected one. |

| Page Down | Scrolls down to the next page and set focus to the first item when popup list opens. |

| Page Up | Scrolls up to the previous page and set focus to the first item when popup list opens. |

| Enter | Selects the focused item, and popup list closes when it is in open state. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Shift + tab | Focuses on the previous TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Alt + Down | Opens the popup list. |

| Alt + Up | Closes the popup list. |

| Esc(Escape) | Closes the popup list when it is in an open state and the currently selected item remains the same. |

| Home | set focus to the first item. |

| End | set focus to the last item. |

In the below sample, focus the MultiSelect component using alt+t keys.

APP.VUE

```
<template>
  <div id="app">
```

```
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' ref='multiselectInstance'
:dataSource='sportsData' :fields='fields' popupHeight="200px"
placeholder="Find a game"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
export default {
  mounted () {
    let multiselectObj = this.$refs.multiselectInstance;
      document.onkeyup = function (e) {
          if (e.altKey && e.keyCode === 84) {
            // press alt+t to focus the control.
            multiselectObj.ej2Instances.focusIn();
          }
      }
      document.onclick = function (e) {
        document.getElementsByClassName('e-
multiselect')[0].classList.remove('e-input-focus');
      }
  },
  data (){
    return {
      sportsData: [
        { id: 'Game1', game: 'Badminton' },
        { id: 'Game2', game: 'Basketball' },
        { id: 'Game3', game: 'Cricket' },
        { id: 'Game4', game: 'Football' },
        { id: 'Game5', game: 'Golf' },
        { id: 'Game6', game: 'Hockey' },
        { id: 'Game7', game: 'Rugby' },
        { id: 'Game8', game: 'Snooker' },
        { id: 'Game9', game: 'Tennis' }
      ],
      fields : { text: 'game', value: 'id' }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/accessibility-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Ensuring accessibility

The MultiSelect component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the MultiSelect component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the MultiSelect component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/multi-select.html" %}

See also

- Accessibility in Syncfusion Vue components

Two way binding in Vue Multi select component

Two-way binding can be achieved by using the v-model directive in Vue. In the following sample, when you change the value in one MultiSelect component, v-model will automatically update the value in the other MultiSelect.

The following example demonstrates how to set the two-way-binding in the MultiSelect.

APP.VUE

```
<template>
<div id="app">
    <div id="wrapper1">
          <ejs-multiselect id='first':dataSource='sportsData'
:fields='fields' :placeholder="waterMark" v-model="value" ></ejs-
multiselect>
    </div>
    <div id="wrapper2">
          <ejs-multiselect id='second' :dataSource='sportsData'
:fields='fields' :placeholder="waterMark" v-model="value" ></ejs-
multiselect>
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from '@syncfusion/ej2-vue-dropdowns';
Vue.use(MultiSelectPlugin);
export default {
  name: 'app',
   data () {
    return {
      waterMark : 'Select a game',
       value: null,
      sportsData: [
     { Id: 'Game1', Game: 'Badminton' },
    { Id: 'Game2', Game: 'Basketball' },
    { Id: 'Game3', Game: 'Cricket' },
    { Id: 'Game4', Game: 'Football' },
    { Id: 'Game5', Game: 'Golf' },
    { Id: 'Game6', Game: 'Hockey' },
    { Id: 'Game7', Game: 'Rugby' },
    { Id: 'Game8', Game: 'Snooker' }
```

```
    ],
    fields: { value: 'Game' }
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
#wrapper1{
  min-width: 250px;
    float: left;
    margin-left: 100px;
}
#wrapper2{
  min-width: 250px;
    float: right;
     margin-right: 100px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/two-way-cs1" %}

How To

Icons support in Vue Multi select component

You can render **icons** to the list items by mapping the iconCss field. This iconCss field create a span in the list item with mapped class name to allow styling as per your need.

In the following sample, icon classes are mapped with iconCss field.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:250px;">
        <br>
        <ejs-multiselect id='multiselect' :dataSource='sortFormatData'
:fields='fields' placeholder="Select a format"></ejs-multiselect>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { MultiSelectPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MultiSelectPlugin);
export default {
  data (){
    return {
      sortFormatData: [
        { Class: 'sort', Type: 'Sort A to Z', Id: '1' },
        { Class: 'filter', Type: 'Filter', Id: '2' },
        { Class: 'clear', Type: 'Clear', Id: '3' }
      ],
      fields : { text: 'Type', iconCss: 'Class', value: 'Id' }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
 .e-list-icon{
    line-height: 1.3;
    padding-right: 10px;
    text-indent: 5px;
}
.sort:before {
    content: '\e890';
    font-family: 'e-icons';
    font-size: 15px;
}
.filter:before {
    content: '\e7ee';
    font-family: 'e-icons';
    font-size: 15px;
    opacity: 0.78;
}
.clear:before {
    content: '\e7fc';
    font-family: 'e-icons';
    font-size: 15px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/multi-select/how-to-cs1" %}

NumericTextbox

Getting Started with the Vue Numerictextbox Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Numerictextbox component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

Copyright © 2001 -2024 Syncfusion Inc.

npm run serve

`

or

`bash

yarn global add @vue/cli

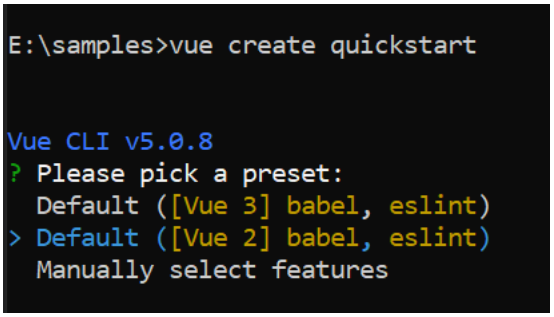
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Numerictextbox component as an example. Install the @syncfusion/ej2-vue-inputs package by running the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

Copyright © 2001 -2024 Syncfusion Inc.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Numerictextbox component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

If you want to refer the combined component styles, please make use of our CRG (Custom Resource Generator) in your application.

Add Syncfusion Vue component

Follow the below steps to add the Vue Numerictextbox component using Composition API or Options API:

1\. First, import and register the Numerictextbox component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { NumericTextBoxComponent as EjsNumerictextbox } from
"@syncfusion/ej2-vue-inputs";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { NumericTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
export default {
components: {
'ejs-numerictextbox': NumericTextBoxComponent
}
}
</script>
```

2\. In the template section, define the Numerictextbox component with the value property.

~/SRC/APP.VUE

```
<template>
<div id="app">
<div class='wrap'>
<ejs-numerictextbox value="10"></ejs-numerictextbox>
</div>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

Copyright © 2001 -2024 Syncfusion Inc.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-numerictextbox value="10"></ejs-numerictextbox>
    </div>
  </div>
</template>
<script setup>
import { NumericTextBoxComponent as EjsNumerictextbox } from
"@syncfusion/ej2-vue-inputs";
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-numerictextbox value="10"></ejs-numerictextbox>
    </div>
  </div>
</template>
<script>
import { NumericTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
export default {
  components: {
    'ejs-numerictextbox': NumericTextBoxComponent
  },
  data () {
    return {
    }
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

Run the project

To run the project, use the following command:

Copyright © 2001 -2024 Syncfusion Inc.

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/numeric-textbox/getting-started/getting-started-cs2" %}

Range validation

You can set the minimum and maximum range of values in the NumericTextBox using the min and max properties, so the numeric value should be in the min and max range.

The validation behavior depends on the strictMode property.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-numerictextbox id="numeric" ref="numeric_instance"
:value="value" :step="step" :min="min" :max="max"></ejs-numerictextbox>
    </div>
  </div>
</template>
<script setup>
import { NumericTextBoxComponent as EjsNumerictextbox } from
"@syncfusion/ej2-vue-inputs";
const min = 1;
const max = 100;
const value = 30;
const step = 2;
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-numerictextbox id="numeric" ref="numeric_instance"
:value="value" :step="step" :min="min" :max="max"></ejs-numerictextbox>
    </div>
  </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import { NumericTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
export default {
  components: {
    'ejs-numerictextbox': NumericTextBoxComponent
  },
  data () {
    return {
        min: 1,
        max: 100,
        value: 30,
        step: 2
    }
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/getting-started/range-validation-cs1" %}

Formatting the value

User can set the format of the NumericTextBox component using format

property. The value will be displayed in the specified format, when the component is in focused out state. For more information about formatting the value, refer to this link.

The below example demonstrates format the value by using currency format value c2.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-numerictextbox id="numeric" ref="numeric_instance"
:format="format" :value="value"></ejs-numerictextbox>
    </div>
  </div>
</template>
<script setup>
import { NumericTextBoxComponent as EjsNumerictextbox } from
"@syncfusion/ej2-vue-inputs";
const format = 'c2';
// sets value to the NumericTextBox
const value = 10;
</script>
\<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-numerictextbox id="numeric" ref="numeric_instance"
:format="format" :value="value"></ejs-numerictextbox>
    </div>
  </div>
</template>
<script>
import { NumericTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
export default {
  components: {
    'ejs-numerictextbox': NumericTextBoxComponent
  },
  data () {
    return {
       format: 'c2',
        // sets value to the NumericTextBox
        value: 10
    }
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/getting-started/formating-value-cs1" %}

Precision of numbers

You can restrict the number of decimals to be entered in the NumericTextBox by using the decimals and validateDecimalOnType properties.

So, you can't enter the number whose precision is greater than the mentioned decimals.

- If validateDecimalOnType is false, number of decimals will not be restricted.

Else, number of decimals will be restricted while typing in the NumericTextBox.

Copyright © 2001 -2024 Syncfusion Inc.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-numerictextbox id="strict" :validateDecimalOnType='true'
placeholder="ValidateDecimalOnType enabled" floatLabelType="Auto"
format='n3' :value="value" :decimals="decimals"></ejs-numerictextbox>
        </div>
        <div class='wrap'>
           <ejs-numerictextbox id="allow" placeholder="ValidateDecimalOnType
disabled" floatLabelType="Auto" format='n3' :value="value"
:decimals="decimals"></ejs-numerictextbox>
        </div>
  </div>
</template>
<script setup>
import { NumericTextBoxComponent as EjsNumerictextbox } from
"@syncfusion/ej2-vue-inputs";
// sets number of decimal places to be allowed by the NumericTextBox
const decimals = 3;
const value = 10;
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
    margin: 35px auto;
    width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-numerictextbox id="strict" :validateDecimalOnType='true'
placeholder="ValidateDecimalOnType enabled" floatLabelType="Auto"
format='n3' :value="value" :decimals="decimals"></ejs-numerictextbox>
        </div>
        <div class='wrap'>
           <ejs-numerictextbox id="allow" placeholder="ValidateDecimalOnType
disabled" floatLabelType="Auto" format='n3' :value="value"
:decimals="decimals"></ejs-numerictextbox>
        </div>
  </div>
</template>
<script>
import { NumericTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
export default {
  components: {
    'ejs-numerictextbox': NumericTextBoxComponent
  },
  data () {
    return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        // sets number of decimal places to be allowed by the NumericTextBox
        decimals: 3,
        value: 10,
    }
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
    margin: 35px auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/getting-started/precision-numbers-cs1" %}

Model binding

In NumericTextBox, the value property supports model binding functionality.

The below example demonstrates model binding functionality with the NumericTextBox and HTML input element.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
  <div class='wrap'>
        <div class='e-input-group' style='margin-bottom:30px'>
        <input class='e-input' type='text' v-model='value'
placeholder='Enter a number'  />
        </div>
        <div>
           <ejs-numerictextbox v-model="value" :value="value"></ejs-
numerictextbox>
        </div>
  </div>
  </div>
</template>
<script setup>
import { NumericTextBoxComponent as EjsNumerictextbox } from
"@syncfusion/ej2-vue-inputs";
const value = 10;
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
  margin: 35px auto;
  width: 240px;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
  <div class='wrap'>
        <div class='e-input-group' style='margin-bottom:30px'>
        <input class='e-input' type='text' v-model='value'
placeholder='Enter a number'  />
        </div>
        <div>
           <ejs-numerictextbox v-model="value" :value="value"></ejs-
numerictextbox>
        </div>
  </div>
  </div>
</template>
<script>
import { NumericTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
export default {
  components: {
    'ejs-numerictextbox': NumericTextBoxComponent
  },
  data () {
    return {
      value : 10
    }
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 .wrap {
  margin: 35px auto;
  width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/getting-started/model-cs1" %}

See Also

- How to perform custom validation using FormValidator
- How to customize the UI appearance of the control
- How to customize the spin button’s up and down arrow
- How to customize the step value and hide spin buttons
- How to prevent nullable input in NumericTextBox
- How to maintain trailing zeros in NumericTextBox

Getting Started with the Vue NumericTextBox Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue NumericTextBox component using the Composition API / Options API.

Copyright © 2001 -2024 Syncfusion Inc.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue NumericTextBox component as an example. To use the Vue NumericTextBox component in the project, the @syncfusion/ej2-vue-inputs package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Copyright © 2001 -2024 Syncfusion Inc.

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the NumericTextBox component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue NumericTextBox component using Composition API or Options API:

1.First, import and register the NumericTextBox component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { NumericTextBoxComponent as EjsNumerictextbox } from
"@syncfusion/ej2-vue-inputs";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { NumericTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: "App",
components: {
'ejs-numerictextbox' : NumericTextBoxComponent,
}
}
</script>
```

2.In the template section, define the NumericTextBox component with the dataSource property and column definitions.

~/SRC/APP.VUE

```
<template>
<div class="control_wrapper">
<ejs-numerictextbox value="10"></ejs-numerictextbox>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-numerictextbox value="10"></ejs-numerictextbox>
</div>
</template>
<script setup>
import { NumericTextBoxComponent as EjsNumerictextbox } from
"@syncfusion/ej2-vue-inputs";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.control_wrapper {
margin: 35px auto;
width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-numerictextbox value="10"></ejs-numerictextbox>
</div>
</template>
<script>
import { NumericTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: 'App',
components: {
"ejs-numerictextbox": NumericTextBoxComponent
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.control_wrapper {
margin: 35px auto;
width: 240px;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

Range validation

You can set the minimum and maximum range of values in the NumericTextBox using the min and max properties, so the numeric value should be in the min and max range.

The validation behavior depends on the strictMode property.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrap'>
<ejs-numerictextbox id="numeric" ref="numeric_instance"
:value="data[0].value" :step="data[0].step" :min="data[0].min"
:max="data[0].max"></ejs-numerictextbox>
</div>
</div>
</template>
<script setup>
import { NumericTextBoxComponent as EjsNumerictextbox } from
"@syncfusion/ej2-vue-inputs";
const data = [{ min: 1,
max: 100,
value:30,
step: 2}]
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
margin: 0 auto;
width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrap'>
<ejs-numerictextbox id="numeric" ref="numeric_instance" :value="value"
:step="step" :min="min" :max="max"></ejs-numerictextbox>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</div>
</template>
<script>
import { NumericTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: 'App',
components: {
"ejs-numerictextbox": NumericTextBoxComponent
},
data () {
return {
min: 1,
max: 100,
value: 30,
step: 2
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
margin: 0 auto;
width: 240px;
}
</style>
```

The output will appear as follows:

Formatting the value

User can set the format of the NumericTextBox component using format

property. The value will be displayed in the specified format, when the component is in focused out state. For more information about formatting the value, refer to this link.

The below example demonstrates format the value by using currency format value c2.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrap'>
<ejs-numerictextbox id="numeric" ref="numeric_instance"
:format="data[0].format" :value="data[0].value"></ejs-numerictextbox>
</div>
</div>
</template>
<script setup>
import { NumericTextBoxComponent as EjsNumerictextbox } from
"@syncfusion/ej2-vue-inputs";
```

```
const data = [{ format: 'c2',
// sets value to the NumericTextBox
value: 10}]
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
margin: 0 auto;
width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrap'>
<ejs-numerictextbox id="numeric" ref="numeric_instance" :format="format"
:value="value"></ejs-numerictextbox>
</div>
</div>
</template>
<script>
import { NumericTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: 'App',
components: {
"ejs-numerictextbox": NumericTextBoxComponent
},
data () {
return {
format: 'c2',
// sets value to the NumericTextBox
value: 10
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
margin: 0 auto;
width: 240px;
}
</style>
```

The output will appear as follows:

Copyright © 2001 -2024 Syncfusion Inc.

Precision of numbers

You can restrict the number of decimals to be entered in the NumericTextBox by using the decimals and validateDecimalOnType properties.

So, you can't enter the number whose precision is greater than the mentioned decimals.

*If validateDecimalOnType is false, number of decimals will not be restricted. Else, number of decimals will be restricted while typing in the NumericTextBox.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrap'>
<ejs-numerictextbox id="strict" :validateDecimalOnType='true'
placeholder="ValidateDecimalOnType enabled" floatLabelType="Auto"
format='n3' :value="data[0].value" :decimals="data[0].decimals"></ejs-
numerictextbox>
</div>
<div class='wrap'>
<ejs-numerictextbox id="allow" placeholder="ValidateDecimalOnType disabled"
floatLabelType="Auto" format='n3' :value="data[0].value"
:decimals="data[0].decimals"></ejs-numerictextbox>
</div>
</div>
</template>
<script setup>
import { NumericTextBoxComponent as EjsNumerictextbox } from
"@syncfusion/ej2-vue-inputs";
const data = [{ decimals: 3,
value: 10 }]
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
margin: 35px auto;
width: 240px;
}
</style>
```

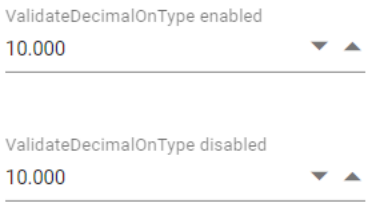
OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrap'>
<ejs-numerictextbox id="strict" :validateDecimalOnType='true'
placeholder="ValidateDecimalOnType enabled" floatLabelType="Auto"
format='n3' :value="value" :decimals="decimals"></ejs-numerictextbox>
</div>
<div class='wrap'>
<ejs-numerictextbox id="allow" placeholder="ValidateDecimalOnType disabled"
floatLabelType="Auto" format='n3' :value="value" :decimals="decimals"></ejs-
numerictextbox>
</div>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import { NumericTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: 'App',
components: {
"ejs-numerictextbox": NumericTextBoxComponent
},
data () {
return {
// sets number of decimal places to be allowed by the NumericTextBox
decimals: 3,
value: 10,
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
margin: 35px auto;
width: 240px;
}
</style>
```

The output will appear as follows:



Model binding

In NumericTextBox, the value property supports model binding functionality. The below example demonstrates model binding functionality with the NumericTextBox and HTML input element.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrap'>
<div class='e-input-group' style='margin-bottom:30px'>
<input class='e-input' type='text' v-model='value' placeholder='Enter a
number'  />
</div>
<ejs-numerictextbox v-model="value" :value="value"></ejs-numerictextbox>
</div>
</div>
</template>
<script setup>
```

```
import { ref } from 'vue'
import { NumericTextBoxComponent as EjsNumerictextbox } from
"@syncfusion/ej2-vue-inputs";
const value = ref(10);
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
margin: 35px auto;
width: 240px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrap'>
<div class='e-input-group' style='margin-bottom:30px'>
<input class='e-input' type='text' v-model='value' placeholder='Enter a
number'  />
</div>
<ejs-numerictextbox v-model="value" :value="value"></ejs-numerictextbox>
</div>
</div>
</template>
<script>
import { NumericTextBoxComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: 'App',
components: {
"ejs-numerictextbox": NumericTextBoxComponent
},
data () {
return {
value : 10
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
margin: 35px auto;
width: 240px;
}
</style>
```

The output will appear as follows:

Copyright © 2001 -2024 Syncfusion Inc.

See Also

- How to perform custom validation using FormValidator
- How to customize the UI appearance of the control
- How to customize the spin button's up and down arrow
- How to customize the step value and hide spin buttons
- How to prevent nullable input in NumericTextBox
- How to maintain trailing zeros in NumericTextBox

Formats in Vue Numerictextbox component

You can format the value of NumericTextBox using format property. The value will be displayed in the specified format when the component is in focused out state. The format string supports both the standard numeric format and custom numeric format.

Standard formats

From the standard numeric formats, you can use the numeric related format specifiers such as n,p and c in the NumericTextBox component. By using these format specifiers, you can achieve the percentage and currency textbox behavior also.

The below example demonstrates percentage and currency formats.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-numerictextbox id="percent" format='p2'
placeholder="Percentage format" floatLabelType="Auto" :value="value"
:step="step" :min="min" :max="max"></ejs-numerictextbox>
        </div>
        <div class='wrap'>
           <ejs-numerictextbox id="currency" placeholder="Currency format"
floatLabelType="Auto" format='c2' :value="currencyvalue"></ejs-
numerictextbox>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(NumericTextBoxPlugin);
export default {
  data () {
    return {
        value: 0.5,
        min: 0,
```

```
        max: 1,
        step: 0.01,
        // sets currency with 2 numbers of decimal places format
        format: 'c2'
        currencyvalue: 10,
    }
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
    margin: 35px auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/number-formats/standard-cs1" %}

Custom formats

From the custom numeric format, you can provide any custom format by combining one or more custom specifiers.

The below examples demonstrate format the value by using currency format string # and 0.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-numerictextbox id="numeric" format='###.##'
placeholder="Custom format string #" floatLabelType="Auto"
:value="value"></ejs-numerictextbox>
        </div>
        <div class='wrap'>
           <ejs-numerictextbox id="numeric1" placeholder="Custom format
string 0" floatLabelType="Auto" format='000.00' :value="numvalue"></ejs-
numerictextbox>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(NumericTextBoxPlugin);
export default {
  data () {
    return {
        // sets the format using custom format string `#`
        value: 10,
        // sets the format using custom format string `0`
        numvalue: 10,
    }
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
    margin: 35px auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/number-formats/custom-cs1" %}

Globalization in Vue Numerictextbox component

Localization

Localization library allows users to localize the default text contents of the NumericTextBox to different cultures using the locale property. In NumericTextBox, spin buttons title for the tooltip will be localized based on the culture.

| Locale key | en-US (default) |

|------|------|

| incrementTitle | Increment value |

| decrementTitle | Decrement value |

Loading translations

To load translation object in your application use `load` function of `L10n` class.

The below example demonstrates the NumericTextBox in `German` culture with the spin buttons tooltip.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-numerictextbox id="numeric" locale='de'
:value="value"></ejs-numerictextbox>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { L10n } from '@syncfusion/ej2-base';
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(NumericTextBoxPlugin);
// Load `German` culture to override spin buttons tooltip text
L10n.load({
    'de': {
        'numerictextbox': {
            incrementTitle: 'Wert erhöhen', decrementTitle: 'Dekrementwert'
        }
    },
});
export default {
```

```
  data () {
    return {
        value: 10,
    }
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
    margin: 0px auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/locale-cs1" %}

Internationalization

Internationalization library provides support for formatting and parsing the number by using the official Unicode CLDR JSON data and also provides the loadCldr method to load the culture specific CLDR JSON data. The NumericTextBox comes with built-in

internationalization support to adapt based on culture. For more information about internationalization, refer to this link.

By default, all the Essential JS 2 component are specific to English culture ('en-US'). If you want to go with the different culture other than English, follow the below steps.

- Install the CLDR-Data package by using the below command (it installs the CLDR JSON data). For more information about CLDR-Data, refer to this link.

`

npm install cldr-data --save

`

Once the package installed, you can find the culture specific JSON data under the location /node_modules/cldr-data.

- Now import the installed CLDR JSON data into the app.vue file.

To import JSON data we need to install the JSON plugin loader. Here we have used the SystemJS JSON plugin loader.

`

npm install systemjs-plugin-json --save-dev

`

Copyright © 2001 -2024 Syncfusion Inc.

- Once installed, configure the system.config.js configuration settings as like below to map the systemjs-plugin-json loader.
- Now import the required culture from the installed location to app.vue file as like the below code snippets.
- Set the culture by using the locale property.

The below example demonstrates the NumericTextBox in German culture with the EUR currency format.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-numerictextbox id="numeric" locale='de' currency='EUR'
format='c2' :value='value'></ejs-numerictextbox>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { loadCldr,L10n,Ajax } from '@syncfusion/ej2-base';
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
import * as numberingSystems from './numberingSystems.json';
import * as currencyData from './currencyData.json';
import * as numbers from './numbers.json';
import * as currencies from './currencies.json';
Vue.use(NumericTextBoxPlugin);
loadCldr(numberingSystems, currencyData, numbers, currencies);
L10n.load({
    'de': {
      'numerictextbox': { incrementTitle: 'Wert erhöhen', decrementTitle:
'Dekrementwert'}
    }
});
export default {
  data (){
      return {
        value : 100
      }
    }
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
    margin: 0px auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/internationalization-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Right to Left(RTL)

Localization library allows users to localize the default text contents of the NumericTextBox to different cultures using the locale property. In NumericTextBox, spin buttons title for the tooltip will be localized based on the culture.

RTL provides an option to switch the text direction and layout of the NumericTextBox component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL NumericTextBox, set the enableRtl to true.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-numerictextbox id="numeric" locale='ar-AE'
floatLabelType='Auto' placeholder='أدخل القيمة' :enableRtl='true'
:value='value'></ejs-numerictextbox>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { loadCldr,L10n,Ajax } from '@syncfusion/ej2-base';
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(NumericTextBoxPlugin);
L10n.load({
    'ar-AE': {
      'numerictextbox': { incrementTitle: 'قيمة الزيادة', decrementTitle:
'قيمة تناقص'}
    }
});
export default {
  data (){
      return {
        value : 100
      }
    }
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
    margin: 0px auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/rtl-cs1" %}

Accessibility in Vue Numerictextbox component

The Numerictextbox component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

Copyright © 2001 -2024 Syncfusion Inc.

The accessibility compliance for the Numerictextbox component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The NumericTextBox characterized with complete ARIA Accessibility support which helps to accessible

by on-screen readers and other assistive technology devices. This component designed with the reference of the guidelines document given in WAI ARAI Accessibility practices.

Copyright © 2001 -2024 Syncfusion Inc.

The NumericTextBox uses the spinbutton role and following ARIA properties to its element based on its state.

| **Property** | **Functionality** |

| --- | --- |

| aria-live | The aria-live attribute indicates the priority of updates to a live region |

| aria-valuemin | The aria-valuemin property specifies the minimum allowable range of the NumericTextBox.|

| aria-valuemax | The aria-valuemax property specifies the maximum allowable range of the NumericTextBox. |

| aria-disabled | The aria-disabled property indicates disabled state of the NumericTextBox. |

| aria-readonly | The aria-readonly property indicates the read-only state of the NumericTextBox. |

| aria-valuenow | The aria-valuenow property specifies the current value of the NumericTextBox. |

| aria-invalid | The aria-invalid property indicates that the user input is incorrect or not within acceptable ranges. |

| aria-label | The aria-label property indicates a string value that labels the NumericTextBox. |

Keyboard interaction

Keyboard interaction of the NumericTextBox component has been designed based on WAI-ARIA Practices described for the NumericTextBox and it is an alternative to mouse actions to interact with the NumericTextBox.

The below table shows shortcut keys and its corresponding usage.

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Arrow Down | Increments the value. |

| Arrow Up | Decrements the value |

APP.VUE

```
<template>
  <div id="app">
    <div class='wrap'>
        <ejs-numerictextbox value="10"></ejs-numerictextbox>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(NumericTextBoxPlugin);
export default {
  data () {
    return {
    }
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/getting-started/getting-started-cs1" %}

Ensuring accessibility

The NumericTextBox component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the NumericTextBox component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the NumericTextBox component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/numeric-textbox.html" %}

See also

- Accessibility in Syncfusion Vue components

Two way binding in Vue Numerictextbox component

Two-way binding can be achieved by using the v-model directive in Vue. In the following sample, when you change the value in one NumericTextBox component, v-model will automatically update the value in the other NumericTextBox.

The following example demonstrates how to set the two-way-binding in the NumericTextBox.

APP.VUE

```
<template>
<div id="app">
    <div id="wrapper1">
        <ejs-numerictextbox floatLabelType="Auto" placeholder="Enter a
value" v-model="value"></ejs-numerictextbox>
    </div>
    <div id="wrapper2">
        <ejs-numerictextbox floatLabelType="Auto" placeholder="Enter a
value" v-model="value"></ejs-numerictextbox>
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { NumericTextBoxPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(NumericTextBoxPlugin);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
 data: function(){
        return {
            value: null
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#wrapper1{
    min-width: 250px;
    float: left;
    margin-left: 100px;
}
#wrapper2{
    min-width: 250px;
    float: right;
    margin-right: 100px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/two-way-cs1" %}

Style appearance in Vue Numerictextbox component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of NumericTextBox wrapper element

Use the following CSS to customize the appearance of wrapper element.

`

/ To specify height and font size /

.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input, .e-input-group textarea.e-input, .e-input-group.e-control-wrapper textarea.e-input {

height: 40px;

font-size: 20px;

}

`

Customizing the NumericTextBox icons

Use the following CSS to customize the Numeric TextBox icons

`

/ To specify font size and background color /

.e-numeric.e-control-wrapper.e-input-group .e-input-group-icon {

font-size: 20px;

background-color: beige;

Copyright © 2001 -2024 Syncfusion Inc.

}
`

How To

Customize the ui appearance of the control in Vue Numerictextbox component

You can change the appearance of the NumericTextBox by adding custom cssClass to the component and enabling styles. Refer to the following example to change the appearance of the NumericTextBox.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-numerictextbox id="numeric" :value="value"
placeholder='Enter value' floatLabelType='Always' cssClass='e-style'></ejs-
numerictextbox>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(NumericTextBoxPlugin);
export default {
  data () {
    return {
      value: 10
    }
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
  margin: 0 auto;
  width: 240px;
  padding-top: 100px;
}
  .e-numeric.e-style .e-control.e-numerictextbox  {
    color: royalblue ;
    font-size: xx-large ;
    border: 0px ;
  }
  .e-input-group.e-control-wrapper:not(.e-success):not(.e-warning):not(.e-
error):not(.e-float-icon-left), .e-float-input.e-control-
wrapper:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-
disabled):not(.e-float-icon-left) {
    border-color: royalblue;
  }
  .e-control-wrapper.e-numeric.e-float-input.e-style .e-spin-down  {
    color:royalblue;
  }
  .e-control-wrapper.e-numeric.e-float-input.e-style .e-float-line::before {
    background: royalblue ;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  }
  .e-control-wrapper.e-numeric.e-float-input.e-style .e-float-line::after {
    background: royalblue ;
   }
  .e-control-wrapper.e-numeric.e-float-input.e-style .e-spin-up {
    color:royalblue ;
  }
  .e-control-wrapper.e-numeric.e-float-input.e-style .e-float-text.e-label-
top {
    color: royalblue ;
    font-size: medium ;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/how-to/cssClass-cs1" %}

Customize the spin buttons up and down arrow in Vue Numerictextbox component

This section explains about how to change/customize spin up and down icons. You can customize spin button icons using e-spin-up and e-spin-down

classes of those buttons.

You can override the default icons of e-spin-up and e-spin-down classes using the following CSS code snippets.

`

.e-numeric .e-input-group-icon.e-spin-up:before {

content: "\e823";

color: rgba(0, 0, 0, 0.54);

}

.e-numeric .e-input-group-icon.e-spin-down:before {

content: "\e934";

color: rgba(0, 0, 0, 0.54);

}

`

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-numerictextbox id="numeric" :value="value"></ejs-
numerictextbox>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(NumericTextBoxPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data () {
    return {
        // sets number of decimal places to be allowed by the NumericTextBox
        value: 10,
    }
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
  margin: 0 auto;
  width: 240px;
  padding-top: 100px;
}
/* csslint ignore:start */
.e-numeric .e-input-group-icon.e-spin-up:before {
    content: "\e823";
    color: rgba(0, 0, 0, 0.54);
}
.e-numeric .e-input-group-icon.e-spin-down:before {
    content: "\e934";
    color: rgba(0, 0, 0, 0.54);
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/how-to/spin-cs1" %}

Customize the step value and hide spin buttons in Vue Numerictextbox component

The spin buttons allow you to increase or decrease the value with the predefined step value. The visibility of spin buttons can be set using theshowSpinButton property.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-numerictextbox id="numeric" :showSpinButton='false'
:value="value" :step="step" :min="min" :max="max"></ejs-numerictextbox>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(NumericTextBoxPlugin);
export default {
  data () {
    return {
        step: 2,
        min: 10,
        max: 100,
        value: 16
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
  margin: 0 auto;
  width: 240px;
  padding-top: 100px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/how-to/step-value-cs1" %}

Maintain trailing zeros in numerictextbox in Vue Numerictextbox component

By default, trailing zeros disappear when the NumericTextBox gets focus. However, you can use the following sample to maintain the trailing zeros while focusing the NumericTextBox.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-numerictextbox id="numeric" :value='value' format='n2'
:decimals='decimals' placeholder='NumericTextBox' :change='onChange'
:created='onCreate'></ejs-numerictextbox>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(NumericTextBoxPlugin);
export default {
  data () {
    return {
      value : 10,
      decimals : 2
    }
  },
  methods: {
    onChange: function (event) {
      var numericObj = document.getElementById("numeric").ej2_instances[0];
      numericObj.element.value =
numericObj.formattedValue(numericObj.decimals, +numericObj.element.value);
    },
     onCreate: function (event) {

document.getElementById("numeric").addEventListener('focus',function(){
        var numericObj =
document.getElementById("numeric").ej2_instances[0];
        numericObj.element.value =
numericObj.formattedValue(numericObj.decimals, +numericObj.element.value);
```

```
      });
    }
}
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
  margin: 0 auto;
  width: 240px;
  padding-top: 100px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/how-to/null-cs1" %}

Perform custom validation using form validator in Vue Numerictextbox component

This section explains how to perform custom validation on the NumericTextBox using FormValidator. The NumericTextBox will be validated when the value changes or the user clicks the submit button.

Validation can be performed by adding custom validation in the rules collection of the FormValidator.

APP.VUE

```
<template>
  <div id="app">
    <div class='wrap'>
      <form id="form-element" class="form-horizontal">
        <div class="form-group">
          <ejs-numerictextbox id="numericRange" name="numericRange"
class="form-control" floatLabelType='Always' :min='min' :max='max'
:strictMode='strictMode'  :created='onCreate' :change='onChange'></ejs-
numerictextbox>
          <button type="button" id="submit_btn" style="margin-top: 40px" v-
on:click="onClick">Submit</button>
        </div>
      </form>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
Vue.use(NumericTextBoxPlugin);
export default {
  data () {
     // checks the value of NumericTextbox and returns the corresponding
boolean value
      var customFn = function(args) {
         var numeric =
document.getElementById('numericRange').ej2_instances[0];
          if(numeric.value>=10 && numeric.value<=100) {
              return true;
```

```
            }
            else {
                return false;
            }
        };
        // sets required property in the FormValidator rules collection
        var options = {
            rules: {
                    'numeric': { required: [true, "Number is required"] },
            },
        }
        // defines FormValidator to validate the NumericTextBox
        var formObject = new FormValidator('#form-element', options);
        // places error label outside the NumericTextBox using the
customPlacement event of FormValidator
        formObject.customPlacement = function (element, errorElement) {

element.parentNode.parentNode.parentNode.appendChild(errorElement);
        };
        //rules for validating the NumericTextbox
        formObject.addRules('numeric', { range: [customFn, "Please enter a
number between 10 to 100"] })
      return {
        min : 10,
        max : 100,
        strictMode : false
      }
   },
   methods: {
     onCreate: function (event){
         document.getElementById("numericRange").setAttribute("name",
"numeric");
     },
     onClick: function (event) {
        // checks the value of NumericTextbox and returns the corresponding
boolean value
        var customFn = function(args) {
           var numeric =
document.getElementById('numericRange').ej2_instances[0];
            if(numeric.value>=10 && numeric.value<=100) {
                return true;
            }
            else {
                return false;
            }
        };
        // sets required property in the FormValidator rules collection
        var options = {
            rules: {
                    'numeric': { required: [true, "Number is required"] },
            },
          }
        // defines FormValidator to validate the NumericTextBox
        var formObject = new FormValidator('#form-element', options);
        // places error label outside the NumericTextBox using the
customPlacement event of FormValidator
        formObject.customPlacement = function (element, errorElement) {
```

```
element.parentNode.parentNode.parentNode.appendChild(errorElement);
      };
      //rules for validating the NumericTextbox
      formObject.addRules('numeric', { range: [customFn, "Please enter a
number between 10 to 100"] });
      formObject.validate("numeric");
       var ele = document.getElementById('numericRange').ej2_instances[0];
       // checks for incomplete value and alerts the formt submit
       if (ele.value >= 10 && ele.value <= 100) {
          alert("Submitted");
       }
    },
    onChange: function (args) {
        // checks the value of NumericTextbox and returns the corresponding
boolean value
      var customFn = function(args) {
         var numeric =
document.getElementById('numericRange').ej2_instances[0];
          if(numeric.value>=10 && numeric.value<=100) {
              return true;
          }
          else {
              return false;
          }
      };
      // sets required property in the FormValidator rules collection
      var options = {
          rules: {
                  'numeric': { required: [true, "Number is required"] },
          },
      }
      // defines FormValidator to validate the NumericTextBox
      var formObject = new FormValidator('#form-element', options);
      // places error label outside the NumericTextBox using the
customPlacement event of FormValidator
      formObject.customPlacement = function (element, errorElement) {

element.parentNode.parentNode.parentNode.appendChild(errorElement);
      };
      //rules for validating the NumericTextbox
      formObject.addRules('numeric', { range: [customFn, "Please enter a
number between 10 to 100"] });
      var  numeric = document.getElementById('numericRange');
      if (numeric.value != null) {
        formObject.validate("numeric");
      }
    }
}
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
 .wrap {
    margin: 0 auto;
```

```
    width: 240px;
    padding-top: 100px;
}
   label.e-error {
      margin-top: -50px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/how-to/validation-cs1" %}

Prevent nullable input in numerictextbox in Vue Numerictextbox component

By default, the value of the NumericTextBox sets to null. In some applications, the value of the NumericTextBox should not be null at any instance. In such cases, following sample can be used to prevent nullable input in NumericTextBox.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-numerictextbox id="numeric" :value='value'
placeholder='NumericTextBox' :created='onCreate' :blur='onBlur'
floatLabelType='Always'></ejs-numerictextbox>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(NumericTextBoxPlugin);
export default {
  data () {
    return {
      value : null
    }
  },
  methods: {
    onCreate: function (event) {
      var numericObj=document.getElementById("numeric").ej2_instances[0];
      //  prevents nullable value during initialization
      if (numericObj.value==null) {
        numericObj.value=0;
      }
   }
   onBlur: function(args) {
     if(args.value == null) {
       numeric.value = 0;
     }
   }
}
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  .wrap {
   margin: 0 auto;
   width: 240px;
   padding-top: 100px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/numeric-textbox/how-to/null-cs2" %}

Pager

Getting started

This section explains you the steps required to create a simple pager

and demonstrate the basic usage of the pager component in Vue environment.

Prerequisites

System requirements for Syncfusion Vue UI components

Setup Vue Environment

You can use Vue CLI to setup your vue applications.

To install Vue CLI use the following commands.

`bash

npm install -g @vue/cli

npm install -g @vue/cli-init

`

Create a Vue Application

Start a new Vue application using below Vue CLI command.

`bash

vue init webpack-simple quickstart

cd quickstart

npm install

`

Adding Syncfusion Pager package

All the available Essential JS 2 packages are published in npmjs.com registry.

To install pager component, use the following command

`bash

npm install @syncfusion/ej2-vue-grids --save

`

The **--save** will instruct NPM to include the grid package inside of the dependencies section of the package.json.

Copyright © 2001 -2024 Syncfusion Inc.

Registering Pager Component

You can register the pager component in your application by using the Vue.use().

Refer to the code example given below.

`ts

import { PagerPlugin } from '@syncfuion/ej2-vue-grids';

Vue.use(PagerPlugin);

`

Adding CSS Reference

The pager has different themes. They are:

- Material
- Office 365
- Fabric
- Bootstrap
- High Contrast

Import pager component CSS as given below in <style> section of the App.vue file.

`

<style>

<!-- Material theme used for this sample -->

@import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";

</style>

`

Adding Pager component

Add the Vue pager by using <ejs-pager> selector in <template> section of the App.vue file.

Import pager plugin into Vue application[src/App.vue] from the package @syncfusion/ej2-vue-grids.

Now place the below pager code in the App.vue.

Here the pager is rendered with totalRecordsCount which is used to render numeric container.

`

<template>

<ejs-pager> </ejs-pager>

</template>

<script>

import Vue from 'vue';

Copyright © 2001 -2024 Syncfusion Inc.

```
import { PagerPlugin } from '@syncfusion/ej2-vue-grids';
Vue.use(PagerPlugin);
export default {
data () {
return {
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
</style>
`
```

Modify the template in [src/app.vue] file to render the pager component.

```
`
<template>
<div id="app">
<ejs-pager :totalRecordsCount='20'> </ejs-pager>
</div>
</template>
<script>
import Vue from 'vue';
import { PagerPlugin } from '@syncfusion/ej2-vue-grids';
Vue.use(PagerPlugin);
export default {
data () {
return {
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

</style>
`

Page Size

pageSize value defines the number of records to be displayed per page. The default value for the pageSize is 12.

APP.VUE

```
<template>
  <div id="app">
      <ejs-pager :pageSize='1' :totalRecordsCount='20'> </ejs-pager>
  </div>
</template>
<script>
import Vue from 'vue';
import { PagerPlugin } from '@syncfusion/ej2-vue-grids';
Vue.use(PagerPlugin);
export default {
  data () {
    return {
    }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pager/pager-cs1" %}

Page Count

pageCount value defines the number of pages to be displayed in the pager component for navigation.

The default value for pageCount is 10 and value will be updated based on totalRecordsCount

and pageSize values.

APP.VUE

```
<template>
  <div id="app">
      <ejs-pager :pageSize='1' :pageCount='3' :totalRecordsCount='20'>
</ejs-pager>
  </div>
</template>
<script>
import Vue from 'vue';
import { PagerPlugin } from '@syncfusion/ej2-vue-grids';
Vue.use(PagerPlugin);
export default {
  data () {
    return {
    }
  }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pager/pager-cs2" %}

Run the application

Use the following command to run the application in browser.

`javascript

npm run dev

`

Output will be appears as follows.

APP.VUE

```
<template>
  <div id="app">
      <ejs-pager :pageSize='1' :pageCount='3' :totalRecordsCount='20'>
</ejs-pager>
  </div>
</template>
<script>
import Vue from 'vue';
import { PagerPlugin } from '@syncfusion/ej2-vue-grids';
Vue.use(PagerPlugin);
export default {
  data () {
    return {
    }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pager/pager-cs3" %}

Style and appearance in Vue Pager component

To modify the Pager appearance, you need to override the default CSS of Pager. Please find the CSS structure that can be used to modify the Pager appearance. Also, you have an option to create your own custom theme for all the JavaScript controls using our Theme Studio.

Customizing the Pager

Use the below CSS to customize the Pager root element.

`

.e-pager {

background-color: #deecf9;

Copyright © 2001 -2024 Syncfusion Inc.

```
}
```

Customizing the Pager container element

Use the below CSS to customize the pager container element.

```
.e-pager .e-pagercontainer {
background-color: #deecf9;
}
```

Customizing the Pager navigation elements

Customize the pager navigation elements, using the below selector.

```
.e-pager .e-prevpagedisabled,
.e-pager .e-prevpage,
.e-pager .e-nextpage,
.e-pager .e-nextpagedisabled,
.e-pager .e-lastpagedisabled,
.e-pager .e-lastpage,
.e-pager .e-firstpage,
.e-pager .e-firstpagedisabled {
background-color: #deecf9;
}
```

Customizing the Pager page numeric link elements

Use the below CSS to customize the pager current page numeric link elements.

```
.e-pager .e-numericitem {
border-radius: initial;
}
```

Customizing the Pager current page numeric element

Using this CSS, you can customize the pager current page numeric item.

```
.e-pager .e-currentitem {
```

Copyright © 2001 -2024 Syncfusion Inc.

background-color: #0078d7;

}

`

Accessibility in Vue Pager component

Accessibility is achieved in the Pager component through WAI-ARIA standard and keyboard navigations. The Pager features can be effectively accessed through assistive technologies such as screen readers.

WAI-ARIA

WAI-ARIA (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components.

The following ARIA attributes are used in the Pager:

- pager (data-role)
- aria-selected (attribute)
- aria-owns (attribute)
- aria-label (attribute)

Keyboard navigation

Pager functionalities can be interactive with keyboard shortcuts.

The following keyboard shortcuts are supported by the Pager.

Interaction Keys | Description

Pager||

Alt + J | Focus on the first pager item.

Tab / Shift + Tab | Focus on the next/previous pager item.

Enter / Space | Select the currently focused page.

Right Arrow / PageDown | Navigate to next page.

Left Arrow / PageUp | Navigate to previous page.

Home / End | Navigate to first and last page.

PDF Viewer

Getting Started

Getting Started with Standalone PDF Viewer component

This section explains the steps to create a simple Standalone Vue PDF Viewer and demonstrate its basic usage.

Prerequisites

System requirements for Syncfusion Vue UI components

Setup Vue environment

You can use Vue CLI to setup your Vue applications.

To install Vue CLI, use the following commands.

`bash

npm install -g @vue/cli

npm install -g @vue/cli-init

`

Create a Vue application

Start a new Vue application using the following Vue CLI command.

`bash

vue init webpack-simple quickstart

cd quickstart

npm install

`

Adding Syncfusion PDF Viewer package

All the available Essential JS 2 packages are published in npmjs.com registry.

- To install PDF Viewer component, use the following command.

`bash

npm install @syncfusion/ej2-vue-pdfviewer --save

`

The **--save** will instruct NPM to include the PDF Viewer package inside the dependencies section of the package.json.

- Create a new directory named 'js' within the 'public' directory using the following command:

`bash

mkdir -p public/js

`

- Duplicate the content of the 'ej2-pdfviewer-lib' folder located in './node_modules/@syncfusion/ej2-pdfviewer/dist' to the public directory using the command:

`bash

cp -R ./node_modules/@syncfusion/ej2-pdfviewer/dist/ej2-pdfviewer-lib public/js/ej2-pdfviewer-lib

`

- Ensure that within your 'public' directory, there exists a directory named 'ej2-pdfviewer-lib' containing the assets of the PDF Viewer library.

Copyright © 2001 -2024 Syncfusion Inc.

- Validate that your server has been configured to utilize the Content-Type: application/wasm MIME type. Additional information can be found in the Troubleshooting section.

Registering PDF Viewer component

You can register the Vue PDF Viewer component in your application by using the Vue.use() as in the below.

```ts
import { PdfViewerPlugin } from '@syncfusion/ej2-vue-pdfviewer';

Vue.use(PdfViewerPlugin);
```

Registering PdfViewerPlugin in Vue, will register the PDF Viewer component along with its required child directives globally.

Adding CSS reference

Add the Vue PDF Viewer component's styles as given below in <style> section of the App.vue file.

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-pdfviewer/styles/material.css";
</style>
```

Adding PDF Viewer component

Add the Vue PDF Viewer by using the <ejs-pdfviewer> selector in <template> section of the App.vue file.

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:resourceUrl="resourceUrl">
```

Copyright © 2001 -2024 Syncfusion Inc.

</ejs-pdfviewer>

</template>

<script>

import Vue from 'vue';

import { PdfViewerPlugin, Toolbar, Magnification, Navigation, LinkAnnotation,

BookmarkView,ThumbnailView, Print,TextSelection, TextSearch,

Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';

Vue.use(PdfViewerPlugin);

export default {

name: 'app',

data () {

return {

documentPath:'https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf',

resourceUrl:"https://cdn.syncfusion.com/ej2/23.1.43/dist/ej2-pdfviewer-lib"

};

},

provide: {

PdfViewer: [Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView,

Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner]

}

}

</script>

`

Run the application

The Vue PDF Viewer application is configured to compile and run the application in a browser. Use the following command to run the application.

`bash

npm run dev

`

Output will be displayed as follows.

APP.VUE

```
<template>
<div id="app">
<ejs-pdfviewer
```

```
id="pdfViewer"
:documentPath="documentPath"
:resourceUrl="resourceUrl">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
resourceUrl:"https://cdn.syncfusion.com/ej2/23.1.43/dist/ej2-pdfviewer-lib",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/pdfviewer/getting-started-cs2" %}

View sample in GitHub

You can refer to our Vue PDF Viewer feature tour page for its groundbreaking feature representations. You can also explore our Vue PDF Viewer example to understand how to explains core features of PDF Viewer.

Limitation over Server-Backed PDF Viewer to Standalone PDF Viewer control

When comparing a Standalone PDF Viewer to a Server-Backed PDF Viewer control, it's crucial to understand the limitations that the Standalone PDF Viewer may have in comparison. These limitations are important to consider

PNG Image Support

The Standalone PDF Viewer does not have the capability to utilize PNG format for adding images to handwritten annotations ,custom stamp ,signature and initial form fields. It's important to be aware that only certain image formats, such as JPEG, are compatible for these purposes.

Local File Access

- The Standalone PDF Viewer control does not have the capability to directly access and load local physical files from a user's device. As a result, it is not possible to use a documentPath to load a PDF file directly from a local server within the viewer.
- The Standalone PDF Viewer allows users to export annotations and form fields from the viewer, it's important to be aware that the viewer does not support the direct import of annotations

Copyright © 2001 -2024 Syncfusion Inc.

and form fields from a locally specified file path. In other words, you can extract annotations and form fields from the viewer, but you cannot reintroduce them into the viewer from external sources by specifying a file path located on your local device.

Note: These limitations are temporary and are expected to be addressed in the near future.

Getting Started with the Vue Pdfviewer Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Pdfviewer component

Prerequisites

System requirements for Syncfusion Vue UI components

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

`

or

`bash

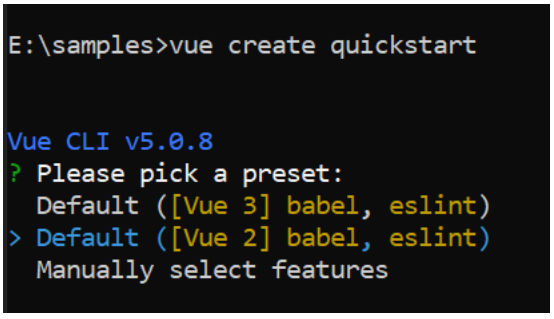
yarn global add @vue/cli

vue create quickstart

cd quickstart

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Pdfviewer component as an example. Install the @syncfusion/ej2-vue-pdfviewer package by running the following command:

`bash

npm install @syncfusion/ej2-vue-pdfviewer --save

`

or

`bash

yarn add @syncfusion/ej2-vue-pdfviewer

`

The **--save** will instruct NPM to include the PDF Viewer package inside the dependencies section of the package.json.

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Pdfviewer component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-pdfviewer/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Pdfviewer component:

1\. First, import and register the Pdfviewer component in the script section of the **src/App.vue** file

~/SRC/APP.VUE

```
<script>
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
export default {
components: {
'ejs-pdfviewer': PdfViewerComponent
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
```

2\. In the template section, define the Pdfviewer component with documentPath and serviceUrl property.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
```

3\. Declare the value for the documentPath and serviceUrl properties in the script section.

~/SRC/APP.VUE

```
<script>
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

~/SRC/APP.VUE

```
<template>
  <div id="app">
    <ejs-pdfviewer
      id="pdfViewer"
      :serviceUrl="serviceUrl"
      :documentPath="documentPath">
    </ejs-pdfviewer>
  </div>
</template>
<script>
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation,
         BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
         Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-
pdfviewer';
export default {
  components: {
    'ejs-pdfviewer': PdfViewerComponent
  },
  name: 'app',
  data () {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    return {

serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
      documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf",
    };
  },
  provide: {
    PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
                 Print, TextSelection, TextSearch, Annotation, FormFields,
FormDesigner ]
  }
}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-vue-
pdfviewer/styles/material.css';
  #pdfViewer {
    height: 640px;
  }
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/pdfviewer/getting-started-cs1" %}

For PDF Viewer serviceUrl creation, follow the steps provided in the link

How to run the PDF Viewer web service

1.Download the sample from the Web service sample in GitHub link. 2.Navigate to the ASP.NET Core folder and open it in the command prompt. 3.Use the below command to restore the required packages.

Copyright © 2001 -2024 Syncfusion Inc.

`

dotnet restore

`

4.Use the below command to run the web service.

`

dotnet run

`

5.You can see that the PDF Viewer server instance runs in the local host with the port number localhost:5001 and navigate to the PDF Viewer Web control localhost:5001/pdfviewer which returns the default get response method. We can bind the link to the serviceUrl property of PDF Viewer as below.

`js

export default {

name: 'app',

data () {

return {

serviceUrl:"https://localhost:5001/pdfviewer",

documentPath:"PDF_Succinctly.pdf"

};

}}

`

Note: When configuring the server-backed PDF viewer, it's essential to understand that there is no need to include the pdfium.js and pdfium.wasm files. Unlike the standalone PDF viewer, which relies on these files for local rendering, the server-backed PDF viewer fetches and renders PDFs directly from the server. Consequently, you can exclude the copy command for deployment process, as they are not required to load and display PDFs in this context.

View sample in GitHub.

You can refer to our Vue PDF Viewer feature tour page for its groundbreaking feature representations. You can also explore our Vue PDF Viewer example to understand how to explains core features of PDF Viewer.

Getting Started with Syncfusion PDF Viewer Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue PDF Viewer component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

Copyright © 2001 -2024 Syncfusion Inc.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue PDF Viewer component as an example. To use the Vue PDF Viewer component in the project, the @syncfusion/ej2-vue-pdfviewer package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-pdfviewer --save

`

or

`bash

yarn add @syncfusion/ej2-vue-pdfviewer

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

Copyright © 2001 -2024 Syncfusion Inc.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the PDF Viewer component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-vue-pdfviewer/styles/material.css';
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue PDF Viewer component using Composition API or Options API:

1.First, import and register the PDF Viewer component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { PdfViewerComponent as EjsPdfviewer, Toolbar, Magnification,
Navigation, LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView,ThumbnailView, Print,TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
}
</script>
```

2. Add the component definition in template section.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
```

Copyright © 2001 -2024 Syncfusion Inc.

```
:documentPath="documentPath"
:style="{ height: '800px', width: '1200px' }">
</ejs-pdfviewer>
</div>
</template>
```

3. Declare the bound properties serviceUrl and documentPath in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { provide } from 'vue';
import { PdfViewerComponent as EjsPdfviewer, Toolbar, Magnification,
Navigation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormDesigner,
FormFields } from '@syncfusion/ej2-vue-pdfviewer';
const serviceUrl =
'https://services.syncfusion.com/vue/production/api/pdfviewer';
const documentPath = 'https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf';
const pdfViewer = null;
provide('PdfViewer', [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields]);
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView,ThumbnailView, Print,TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
export default {
name: 'App',
components: {
"ejs-pdfviewer": PdfViewerComponent
},
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]}
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<ejs-pdfviewer
ref="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:style="{ height: '800px', width: '1200px' }">
</ejs-pdfviewer>
</template>
<script setup>
import { provide } from 'vue';
import { PdfViewerComponent as EjsPdfviewer, Toolbar, Magnification,
Navigation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormDesigner,
FormFields } from '@syncfusion/ej2-vue-pdfviewer';
const serviceUrl =
'https://services.syncfusion.com/vue/production/api/pdfviewer';
const documentPath = 'https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf';
const pdfViewer = null;
provide('PdfViewer', [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields]);
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-vue-pdfviewer/styles/material.css';
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:style="{ height: '800px', width: '1200px' }">
</ejs-pdfviewer>
</template>
<script>
import { PdfViewerComponent, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormDesigner,
FormFields } from '@syncfusion/ej2-vue-pdfviewer';
export default {
name: 'App',
components: {
"ejs-pdfviewer": PdfViewerComponent
},
data() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
return {
serviceUrl: "https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-vue-pdfviewer/styles/material.css';
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

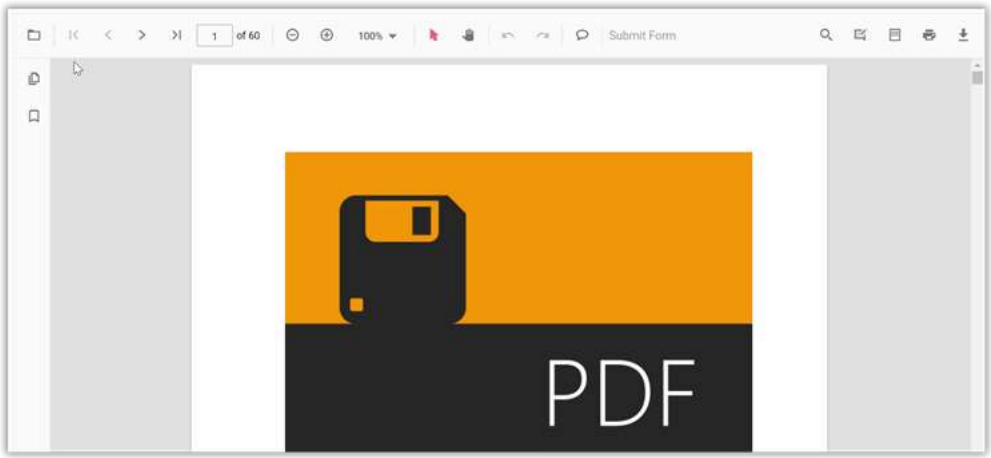
or

`bash

yarn run dev

`

The output will appear as follows:



View sample in GitHub.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

Feature module in Vue Pdfviewer component

The PDF Viewer features are segregated into individual feature-wise modules to enable selectively referencing in the application. The required modules should be injected to extend its functionality. The following are the selective modules of PDF Viewer that can be included as required:

The available PdfViewer modules are:

- **Toolbar**:- Built-in toolbar for better user interaction.
- **Magnification**:- Perform zooming operation for better viewing experience.
- **Navigation**:- Easy navigation across the PDF pages.
- **LinkAnnotation**:- Easy navigation within and outside of the PDF document.
- **ThumbnailView**:- Easy navigation with in the PDF document.
- **BookmarkView**:- Easy navigation based on the bookmark content of the PDF document.
- **TextSelection**:- Select and copy text from a PDF file.
- **TextSearch**:- Search a text easily across the PDF document.
- **Print**:- Print the entire document or a specific page directly from the browser.
- **Annotation**:- Annotations can be added or edited in the PDF document.
- **FormFields**:- Preserve the form fields in the PDF document.
- **FormDesigner**:- Form fields can be added or edited in the PDF document.

In addition to injecting the required modules in your application, enable corresponding properties to extend the functionality for a PDF Viewer instance.

Refer to the following table.

| Module | Property to enable the functionality for a PDF Viewer instance |

|---|---|

|Toolbar|<ejs-pdfviewer :enableToolbar="true" ></ejs-pdfviewer>|

|Magnification|<ejs-pdfviewer :enableMagnification="true" ></ejs-pdfviewer>|

|Navigation|<ejs-pdfviewer :enableNavigation="true" ></ejs-pdfviewer>|

|LinkAnnotation|<ejs-pdfviewer :enableHyperlink="true" ></ejs-pdfviewer>|

|ThumbnailView|<ejs-pdfviewer :enableThumbnail="true" ></ejs-pdfviewer>|

|BookmarkView|<ejs-pdfviewer :enableBookmark="true" ></ejs-pdfviewer>|

|TextSelection|<ejs-pdfviewer :enableTextSelection="true" ></ejs-pdfviewer>|

|TextSearch|<ejs-pdfviewer :enableTextSearch="true" ></ejs-pdfviewer>|

|Print|<ejs-pdfviewer :enablePrint="true" ></ejs-pdfviewer>|

|Annotation|<ejs-pdfviewer :enableAnnotation="true" ></ejs-pdfviewer>|

|FormFields|<ejs-pdfviewer :enableFormFields="true" ></ejs-pdfviewer>|

|FormDesigner|<ejs-pdfviewer :enableFormDesigner="true" ></ejs-pdfviewer>|

Copyright © 2001 -2024 Syncfusion Inc.

See also

- Toolbar items
- Toolbar customization

Open PDF files

You might need to open and view the PDF files from various location. In this section, you can find the information about how to open PDF files from URL, database and as base64 string.

Opening a PDF from URL

If you have your PDF files in the web, you can open it in the viewer using URL.

Step 1: Create a Simple PDF Viewer Sample in Vue

Start by following the steps provided in this link to create a simple PDF viewer sample in Vue. This will give you a basic setup of the PDF viewer component.

Step 2: Modify the `PdfViewerController.cs` File in the Web Service Project

1. Create a web service project in .NET Core 3.0 or above. You can refer to this link for instructions on how to create a web service project.
2. Open the `PdfViewerController.cs` file in your web service project.
3. Modify the `Load()` method to open it in the viewer using URL

```csharp
public IActionResult Load([FromBody] Dictionary<string, string> jsonData)
{
// Initialize the PDF viewer object with memory cache object
PdfRenderer pdfviewer = new PdfRenderer(_cache);
MemoryStream stream = new MemoryStream();
object jsonResult = new object();
if (jsonObject != null && jsonObject.ContainsKey("document"))
{
if (bool.Parse(jsonObject["isFileName"]))
{
string documentPath = GetDocumentPath(jsonData["document"]);
if (!string.IsNullOrEmpty(documentPath))
{
byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
stream = new MemoryStream(bytes);
}
```

```
else
{
string fileName = jsonData["document"].Split(new string[] { "://" }, StringSplitOptions.None)[0];
if (fileName == "http" || fileName == "https")
{
WebClient WebClient = new WebClient();
byte[] pdfDoc = WebClient.DownloadData(jsonData["document"]);
stream = new MemoryStream(pdfDoc);
}
else
{
return this.Content(jsonData["document"] + " is not found");
}
}
}
else
{
byte[] bytes = Convert.FromBase64String(jsonObject["document"]);
stream = new MemoryStream(bytes);
}
}
jsonResult = pdfviewer.Load(stream, jsonObject);
return Content(JsonConvert.SerializeObject(jsonResult));
}
```

Step 3: Set the PDF Viewer Properties in React PDF viewer component

Modify the `serviceUrl` property of the PDF viewer component with the accurate URL of your web service project, replacing `https://localhost:44396/pdfviewer` with the actual URL of your server.Modify the documentPath with the correct PDF Document URL want to load.

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
```

```
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data() {
return {
// Replace the "localhost:44396" with the actual URL of your server
serviceUrl: "https://localhost:44396/pdfviewer",
// Replace  correct PDF Document URL want to load
documentPath: "https://cdn.syncfusion.com/content/PDFViewer/flutter-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner]
}
}
</script>
`
```

View sample in GitHub

Opening a PDF from base64 data

The following code steps how the PDF file can be loaded in PDF Viewer as base64 string.

Step 1: Create a Simple PDF Viewer Sample in Angular

Start by following the steps provided in this link to create a simple PDF viewer sample in Angular. This will give you a basic setup of the PDF viewer component.

Copyright © 2001 -2024 Syncfusion Inc.

Step 2: Use the following code snippet to load the PDF document using a base64 string.

```
<template>
<div id="app">
<button v-on:click="load">LoadDocumentFromBase64</button>
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"

</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data() {
return {
// Replace the "localhost:44396" with the actual URL of your server
serviceUrl: "https://localhost:44396/pdfviewer"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]
},
methods: {
// Event triggers on the Export FDF button click.
```

Copyright © 2001 -2024 Syncfusion Inc.

load: function (event) {

var viewer = document.getElementById('pdfViewer').ej2_instances[0];

viewer.load('data:application/pdfbase64,'+ AddBase64String, null);

}

}

}

</script>

`

View sample in GitHub

Opening a PDF from database

To load a PDF file from SQL Server database in a PDF Viewer, you can follow the steps below

Step 1: Create a Simple PDF Viewer Sample in Vue

Start by following the steps provided in this link to create a simple PDF viewer sample in Vue. This will give you a basic setup of the PDF viewer component.

Step 2: Modify the PdfViewerController.cs File in the Web Service Project

1. Create a web service project in .NET Core 3.0 or above. You can refer to this link for instructions on how to create a web service project.
2. Open the PdfViewerController.cs file in your web service project.
3. Import the required namespaces at the top of the file:

`csharp

using System.IO;

using System.Data.SqlClient;

`

4. Add the following private fields and constructor parameters to the PdfViewerController class, In the constructor, assign the values from the configuration to the corresponding fields

`csharp

private IConfiguration _configuration;

public readonly string _connectionString;

public PdfViewerController(IWebHostEnvironment hostingEnvironment, IMemoryCache cache, IConfiguration configuration)

{

_hostingEnvironment = hostingEnvironment;

_cache = cache;

Copyright © 2001 -2024 Syncfusion Inc.

```
_configuration = configuration;
connectionString = configuration.GetValue<string>("ConnectionString");
}
`
```

5. Modify the Load() method to open it in the viewer using URL

```
`csharp
public IActionResult Load([FromBody] Dictionary<string, string> jsonData)
{
// Initialize the PDF viewer object with memory cache object
PdfRenderer pdfviewer = new PdfRenderer(_cache);
MemoryStream stream = new MemoryStream();
object jsonResult = new object();
if (jsonObject != null && jsonObject.ContainsKey("document"))
{
if (bool.Parse(jsonObject["isFileName"]))
{
string documentPath = GetDocumentPath(jsonData["document"]);
if (!string.IsNullOrEmpty(documentPath))
{
byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
stream = new MemoryStream(bytes);
}
string documentName = jsonObject["document"];
string connectionString = _connectionString;
System.Data.SqlClient.SqlConnection connection = new
System.Data.SqlClient.SqlConnection(connectionString);
//Searches for the PDF document from the database
string query = "SELECT FileData FROM Table WHERE FileName = '" + documentName + "'";
System.Data.SqlClient.SqlCommand command = new System.Data.SqlClient.SqlCommand(query,
connection);
connection.Open();
using (SqlDataReader reader = command.ExecuteReader())
{
```

```
if (reader.Read())
{
byte[] byteArray = (byte[])reader["FileData"];
stream = new MemoryStream(byteArray);
}
}
}
else
{
byte[] bytes = Convert.FromBase64String(jsonObject["document"]);
stream = new MemoryStream(bytes);
}
}
jsonResult = pdfviewer.Load(stream, jsonObject);
return Content(JsonConvert.SerializeObject(jsonResult));
}
```

6. Open the appsettings.json file in your web service project, Add the following lines below the existing “AllowedHosts” configuration

```json
{
"Logging": {
"LogLevel": {
"Default": "Information",
"Microsoft.AspNetCore": "Warning"
}
},
"AllowedHosts": "*",
"ConnectionString": "Your connection string for SQL server"
}
```

Note: Replace **Your Connection string from SQL server** with the actual connection string for your SQL Server database

Copyright © 2001 -2024 Syncfusion Inc.

Note: The **System.Data.SqlClient** package must be installed in your application to use the previous code example. You need to modify the connectionString variable in the previous code example as per the connection string of your database.

View sample in GitHub

Saving PDF file

After editing the PDF file with various annotation tools, you will need to save the updated PDF to the server, database, or local file system.

Save PDF file to Server

Need to save the modified PDF back to a server. To achieve this, proceed with the following steps

Step 1: Create a Simple PDF Viewer Sample in Vue

Start by following the steps provided in this link to create a simple PDF viewer sample in Vue. This will give you a basic setup of the PDF viewer component.

Step 2: Modify the PdfViewerController.cs File in the Web Service Project

1. Create a web service project in .NET Core 3.0 or above. You can refer to this link for instructions on how to create a web service project.
2. Open the PdfViewerController.cs file in your web service project.
3. Modify the Download() method to open it in the viewer using URL

```csharp
public IActionResult Download([FromBody] Dictionary<string, string> jsonObject)
{
//Initialize the PDF Viewer object with memory cache object
PdfRenderer pdfviewer = new PdfRenderer(_cache);
string documentBase = pdfviewer.GetDocumentAsBase64(jsonObject);
MemoryStream stream = new MemoryStream();
string documentName = jsonObject["document"];
string result = Path.GetFileNameWithoutExtension(documentName);
string fileName = result + "_downloaded.pdf";
// Save the file on the server
string serverFilePath = @"Path to where you need to save your file in the server";
string filePath = Path.Combine(serverFilePath, fileName);
using (FileStream fileStream = new FileStream(filePath, FileMode.Create))
{
//Saving the new file in root path of application
stream.CopyTo(fileStream);
fileStream.Close();
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
return Content(documentBase);
}
```

Step 3: Set the PDF Viewer Properties in React PDF viewer component

Modify the `serviceUrl` property of the PDF viewer component with the accurate URL of your web service project, replacing `https://localhost:44396/pdfviewer` with the actual URL of your server.Modify the documentPath with the correct PDF Document URL want to load.

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data() {
return {
// Replace the "localhost:44396" with the actual URL of your server
serviceUrl: "https://localhost:44396/pdfviewer",
documentPath: "https://cdn.syncfusion.com/content/PDFViewer/flutter-succinctly.pdf"
};
},
provide: {
```

```
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner]
}
}
</script>
```

[View sample in GitHub]

Download PDF file as a copy

In the built-in toolbar, you have an option to download the updated PDF to the local file system, you can use it to download the PDF file.

```
<template>
<div id="app">
<button v-on:click="downloadClicked">Download</button>
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data() {
return {
// Replace the "localhost:44396" with the actual URL of your server
serviceUrl: "https://localhost:44396/pdfviewer",
// Replace PDF_Succinctly.pdf with the actual document name that you want to load
```

```
documentPath:"PDF_Succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]
},
methods: {
downloadClicked: function (args) {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.download();
},
}
}
</script>
`
```

Save PDF file to Database

If you have plenty of PDF files stored in database and you want to save the updated PDF file back to the database, use the following code example.

Step 1: Create a Simple PDF Viewer Sample in Vue

Start by following the steps provided in this link to create a simple PDF viewer sample in Vue. This will give you a basic setup of the PDF viewer component.

Step 2: Modify the `PdfViewerController.cs` File in the Web Service Project

1. Create a web service project in .NET Core 3.0 or above. You can refer to this link for instructions on how to create a web service project.
2. Open the `PdfViewerController.cs` file in your web service project.
3. Import the required namespaces at the top of the file:

```
`csharp
using System.IO;
using System.Data.SqlClient;
`
```

4. Add the following private fields and constructor parameters to the `PdfViewerController` class, In the constructor, assign the values from the configuration to the corresponding fields

Copyright © 2001 -2024 Syncfusion Inc.

`csharp

private IConfiguration _configuration;

public readonly string _connectionString;

public PdfViewerController(IWebHostEnvironment hostingEnvironment, IMemoryCache cache, IConfiguration configuration)

{

_hostingEnvironment = hostingEnvironment;

_cache = cache;

_configuration = configuration;

connectionString = configuration.GetValue<string>("ConnectionString");

}

`

5. Modify the Download() method to open it in the viewer using URL

`csharp

[HttpPost("Download")]

[Microsoft.AspNetCore.Cors.EnableCors("MyPolicy")]

[Route("[controller]/Download")]

//Post action for downloading the PDF documents

public async Task<IActionResult> Download([FromBody] Dictionary<string, string> jsonObject)

{

//Initialize the PDF Viewer object with memory cache object

PdfRenderer pdfviewer = new PdfRenderer(_cache);

string documentBase = pdfviewer.GetDocumentAsBase64(jsonObject);

byte[] documentBytes = Convert.FromBase64String(documentBase.Split(",")[1]);

string documentId = jsonObject["documentId"];

string result = Path.GetFileNameWithoutExtension(documentId);

string fileName = result + "_downloaded.pdf";

string connectionString = _connectionString;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand cmd = new SqlCommand("INSERT INTO Table (FileName, fileData) VALUES (@FileName, @fileData)", connection))

```
{
cmd.Parameters.AddWithValue("@FileName", fileName);
cmd.Parameters.AddWithValue("@fileData", documentBytes);
cmd.ExecuteNonQuery();
}
connection.Close();
}
return Content(documentBase);
}
```

6. Open the appsettings.json file in your web service project, Add the following lines below the existing “AllowedHosts” configuration

```json
{
"Logging": {
"LogLevel": {
"Default": "Information",
"Microsoft.AspNetCore": "Warning"
}
},
"AllowedHosts": "*",
"ConnectionString": "Your connection string for SQL server"
}
```

Note: Replace **Your Connection string from SQL server** with the actual connection string for your SQL Server database

Note: The **System.Data.SqlClient** package must be installed in your application to use the previous code example. You need to modify the connectionString variable in the previous code example as per the connection string of your database.

View sample in GitHub

Accessibility in Syncfusion Vue PDF Viewer components

The PDF Viewer component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the PDF Viewer component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

WAI-ARIA (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript,and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components. The following ARIA attributes are used in the PDF Viewer component:

| Attributes | Purpose |

| --- | --- |

| aria-disabled| Indicates whether the PDF Viewer component is in a disabled state or not.|

| aria-expanded| Indicates whether the suggestion list has expanded or not. |

| aria-readonly | Indicates the readonly state of the PDF Viewer element. |

| aria-haspopup | Indicates whether the PDF Viewer input element has a suggestion list or not. |

| aria-label | Indicates the breadcrumb item text. |

| aria-labelledby | Provides a label for the PDF Viewer. Typically, the "aria-labelledby" attribute will contain the id of the element used as the PDF Viewer's title. |

| aria-describedby | This attribute points to the PDF Viewer element describing the one it's set on. |

| aria-orientation | Indicates whether the PDF Viewer element is oriented horizontally or vertically. |

| aria-valuetext | Returns the current text of the PDF Viewer. |

| aria-valuemax | Indicates the Maximum value of the PDF Viewer. |

| aria-valuemin | Indicates the Minimum value of the PDF Viewer. |

| aria-valuenow | Indicates the current value of the PDF Viewer. |

| aria-controls | Attribute is set to the button and it points to the corresponding content. |

Keyboard interaction

The PDF Viewer component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Message component.

| **Press (Windows)** |**Press (Macintosh)** | **To do this** |

| --- | --- | --- |

|||**Shortcuts for page navigation**|

| Home | Function + Left arrow |Navigate to the first page |

| End |Function + Right arrow |Navigate to the last page |

|Up Arrow |Up Arrow |Navigate to the previous page|

| Down Arrow | Down Arrow | Navigate to the next page |

|||**Shortcuts for Zooming**|

|CONTROL + = |COMMAND + = | Perform zoom-in operation |

| CONTROL + - | COMMAND + - | Perform zoom-out operation |

|CONTROL + 0 |COMMAND + 0 | Retain the zoom level to 1 |

|||**Shortcut for Text Search**|

| CONTROL + F | COMMAND + F |Open the search toolbar|

|||**Shortcut for Text Selection**|

Copyright © 2001 -2024 Syncfusion Inc.

|CONTROL + C |CONTROL + C | Copy the selected text or annotation or form field |

| CONTROL + X | CONTROL + X |Cut the selected text or annotation of the form field|

|CONTROL + Y |CONTROL + Y |Paste the selected text or annotation or form field|

|||**Shortcuts for the general operation**|

| CONTROL + Z | CONTROL + Z |Undo the action|

|CONTROL + Y |CONTROL + Y |Redo the action|

| CONTROL + P | CONTROL + P |Print the document|

|Delete |Delete |Delete the annotations and form fields|

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:resourceUrl="resourceUrl">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
resourceUrl:"https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-lib"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

Ensuring accessibility

The PDF Viewer component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the PDF Viewer component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the PDF Viewer component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/pdf-viewer.html" %}

See also

- Accessibility in Syncfusion Vue components

Toolbar in Vue Pdfviewer component

The PDF Viewer comes with a powerful built-in toolbar to execute important actions such as page navigation, text search,view mode,download,print,bookmark, and thumbnails.

The following table shows built-in toolbar items and its actions:-

| Option | Description |

|---|---|

| OpenOption | This option provides an action to load the PDF documents to the PDF Viewer.|

| PageNavigationTool | This option provides an action to navigate the pages in the PDF Viewer. It contains GoToFirstPage,GoToLastPage,GotoPage,GoToNext, and GoToLast tools.|

| MagnificationTool |This option provides an action to magnify the pages either with predefined or user defined zoom factors in the PDF Viewer. Contains ZoomIn, ZoomOut, Zoom, FitPage and FitWidth tools|

| PanTool | This option provides an action for panning the pages in the PDF Viewer.|

Copyright © 2001 -2024 Syncfusion Inc.

| SelectionTool | This option provides an action to enable/disable the text selection in the PDF Viewer.|

| SearchOption | This option provides an action to search a word in the PDF documents.|

| PrintOption | This option provides an action to print the PDF document being loaded in the PDF Viewer.|

| DownloadOption |This Download option provides an action to download the PDF document that has been loaded in the PDF Viewer.|

| UndoRedoTool | This tool provides options to undo and redo the annotation actions performed in the PDF Viewer.|

| AnnotationEditTool | This tool provides options to enable or disable the edit mode of annotation in the PDF Viewer.|

| FormDesignerEditTool | This tool provides options to enable or disable the edit mode of form fields in the PDF Viewer.|

Show/Hide the default toolbar

The PDF Viewer has an option to show or hide the complete default toolbar. You can achieve this by using following two ways.,

- **Show/Hide toolbar using enableToolbar API as in the following code snippet**

APP.VUE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:enableToolbar="false"
:resourceUrl="resourceUrl">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
resourceUrl:"https://cdn.syncfusion.com/ej2/23.1.43/dist/ej2-pdfviewer-lib",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner ]}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
```

Note: To set up the **server-backed PDF Viewer** in the app.vue file, include the following serviceUrl:

serviceUrl: “https://services.syncfusion.com/vue/production/api/pdfviewer”

Within the template, configure the PDF Viewer by adding the :serviceUrl=“serviceUrl” attribute inside the element.

{% previewsample "page.domainurl/code-snippet/pdfviewer/toolbar/toolbar-hide-cs1" %}

- **Show/Hide toolbar using showToolbar as in the following code snippet**

APP.VUE

```
<template>
<div id="app">
<button v-on:click="showToolbarClicked">showToolbarClicked</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad"
:resourceUrl="resourceUrl">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
resourceUrl:"https://cdn.syncfusion.com/ej2/23.1.43/dist/ej2-pdfviewer-lib"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
showToolbarClicked: function (args) {
viewer.toolbar.showToolbar(false);
},
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
},
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
```

Note: To set up the **server-backed PDF Viewer** in the app.vue file, include the following serviceUrl:

serviceUrl: “https://services.syncfusion.com/vue/production/api/pdfviewer”

Within the template, configure the PDF Viewer by adding the :serviceUrl=“serviceUrl” attribute inside the element.

{% previewsample "page.domainurl/code-snippet/pdfviewer/toolbar/toolbar-method-cs1" %}

Show/Hide the default toolbaritem

The PDF Viewer has an option to show or hide these grouped items in the default toolbar.

- **Show/Hide toolbaritem using toolbarSettings as in the following code snippet.**

APP.VUE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:enableTextSelection="false"
:toolbarSettings="toolbarSettings"
:resourceUrl="resourceUrl">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
resourceUrl:"https://cdn.syncfusion.com/ej2/23.1.43/dist/ej2-pdfviewer-lib",
toolbarSettings:{ showTooltip : true, toolbarItems: ['OpenOption',
'UndoRedoTool', 'PageNavigationTool', 'MagnificationTool', 'PanTool',
'SelectionTool', 'CommentTool', 'SubmitForm', 'AnnotationEditTool',
'FormDesignerEditTool', 'FreeTextAnnotationOption', 'InkAnnotationOption',
'ShapeAnnotationOption', 'StampAnnotation', 'SignatureOption',
'SearchOption', 'PrintOption', 'DownloadOption'], annotationToolbarItems:
['HighlightTool', 'UnderlineTool', 'StrikethroughTool', 'ColorEditTool',
'OpacityEditTool', 'AnnotationDeleteTool', 'StampAnnotationTool',
'HandWrittenSignatureTool', 'InkAnnotationTool', 'ShapeTool',
'CalibrateTool', 'StrokeColorEditTool', 'ThicknessEditTool',
'FreeTextAnnotationTool', 'FontFamilyAnnotationTool',
'FontSizeAnnotationTool', 'FontStylesAnnotationTool',
```

```
'FontAlignAnnotationTool', 'FontColorAnnotationTool', 'CommentPanelTool'],
formDesignerToolbarItems: ['TextboxTool', 'PasswordTool', 'CheckBoxTool',
'RadioButtonTool', 'DropdownTool', 'ListboxTool', 'DrawSignatureTool',
'DeleteTool']}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner ]}
}
</script>
```

Note: To set up the **server-backed PDF Viewer** in the app.vue file, include the following serviceUrl:

serviceUrl: “https://services.syncfusion.com/vue/production/api/pdfviewer”

Within the template, configure the PDF Viewer by adding the :serviceUrl=“serviceUrl” attribute inside the element.

{% previewsample "page.domainurl/code-snippet/pdfviewer/toolbar/toolbar-items-cs1" %}

- **Show/Hide toolbaritem using showToolbaritem as in the following code snippet**

APP.VUE

```
<template>
<div id="app">
<button v-on:click="showToolbarClicked">ShowToolbar</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad"
:resourceUrl="resourceUrl">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
resourceUrl:"https://cdn.syncfusion.com/ej2/23.1.43/dist/ej2-pdfviewer-lib"
};
},
provide: {
```

```
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner ]
},
methods: {
showToolbarClicked: function (args) {
viewer.toolbar.showToolbarItem(["OpenOption"],false);
},
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
},
}
}
</script>
```

Note: To set up the **server-backed PDF Viewer** in the app.vue file, include the following serviceUrl:

serviceUrl: “https://services.syncfusion.com/vue/production/api/pdfviewer”

Within the template, configure the PDF Viewer by adding the :serviceUrl=“serviceUrl” attribute inside the element.

{% previewsample "page.domainurl/code-snippet/pdfviewer/toolbar/toolbar-items-method-cs1" %}

See also

- Toolbar customization
- Feature Modules

Navigation in Vue Pdfviewer component

The PDF Viewer supports different internal and external navigations.

Toolbar page navigation option

The default toolbar of PDF Viewer contains the following navigation options

- **Go to page**:- Navigates to the specific page of a PDF document.
- **Show next page**:- Navigates to the next page of PDF a document.
- **Show previous page**:- Navigates to the previous page of a PDF document.
- **Show first page**:- Navigates to the first page of a PDF document.
- **Show last page**:- Navigates to the last page of a PDF document.

You can enable/disable page navigation option in PDF Viewer using the EnableNavigation property and use the following code snippet,

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:enableNavigation="true">
</ejs-pdfviewer>
</div>
```

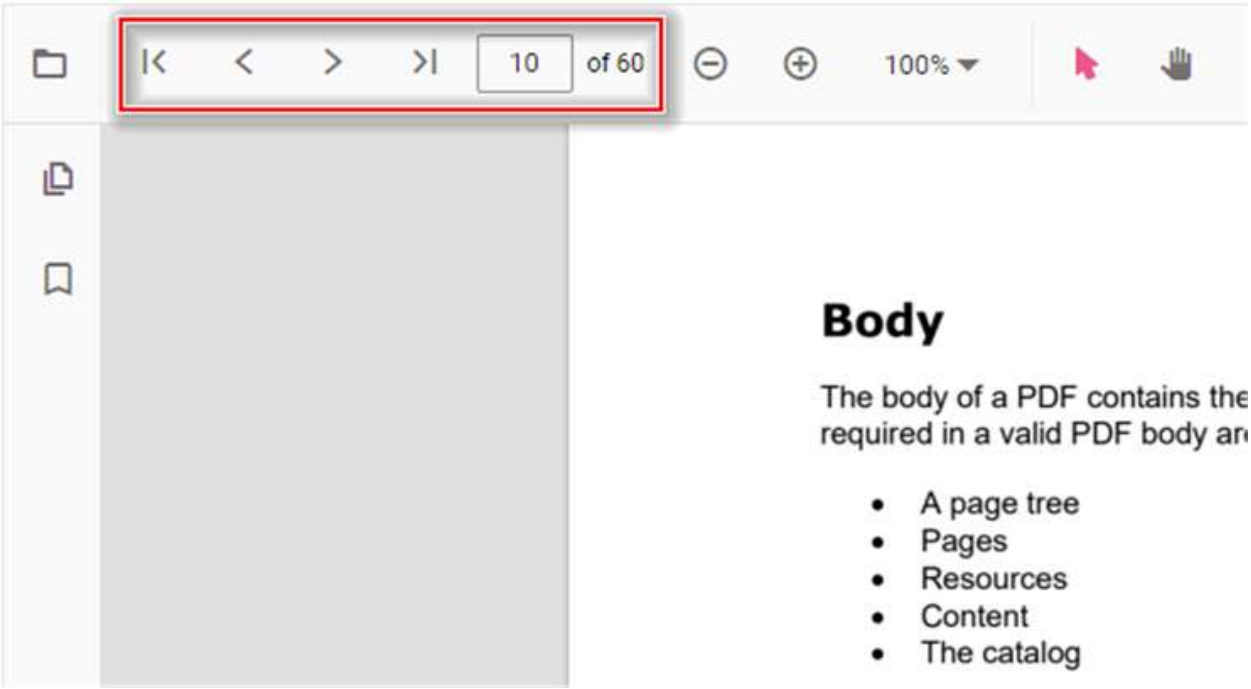
Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:enableNavigation="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.



Also, you can programmatically perform page navigation options as follows.

STANDALONE

```
<template>
<div>
<button v-on:click="goToFirstPage">Go To First Page</button>
<button v-on:click="goToLastPage">Go To last Page</button>
<button v-on:click="goToNextPage">Go To Next Page</button>
<button v-on:click="goToPage">Go To Page</button>
<button v-on:click="goToPreviousPage">Go To Previous Page</button>
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
methods: {
//Go To First Page
goToFirstPage: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.navigation.goToFirstPage();
},
//Go To Last Page
goToLastPage: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.navigation.goToLastPage();
},
//Go To Next Page
goToNextPage: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.navigation.goToNextPage();
},
//Go To Page
goToPage: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.navigation.goToPage(4);
},
//Go To Previous Page
goToPreviousPage: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.navigation.goToPreviousPage();
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div>
<button v-on:click="goToFirstPage">Go To First Page</button>
<button v-on:click="goToLastPage">Go To last Page</button>
<button v-on:click="goToNextPage">Go To Next Page</button>
<button v-on:click="goToPage">Go To Page</button>
<button v-on:click="goToPreviousPage">Go To Previous Page</button>
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
data () {
return {
```

```
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
methods: {
//Go To First Page
goToFirstPage: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.navigation.goToFirstPage();
},
//Go To Last Page
goToLastPage: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.navigation.goToLastPage();
},
//Go To Next Page
goToNextPage: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.navigation.goToNextPage();
},
//Go To Page
goToPage: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.navigation.goToPage(4);
},
//Go To Previous Page
goToPreviousPage: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.navigation.goToPreviousPage();
}
}
}
</script>
```

Find the here to perform the page navigation options programmatically.

Bookmark navigation

The Bookmarks saved in PDF files are loaded and made ready for easy navigation. You can enable/disable bookmark navigation by using the following code snippet.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:enableBookmark="false">
</ejs-pdfviewer>
</div>
</template>
<script>
```

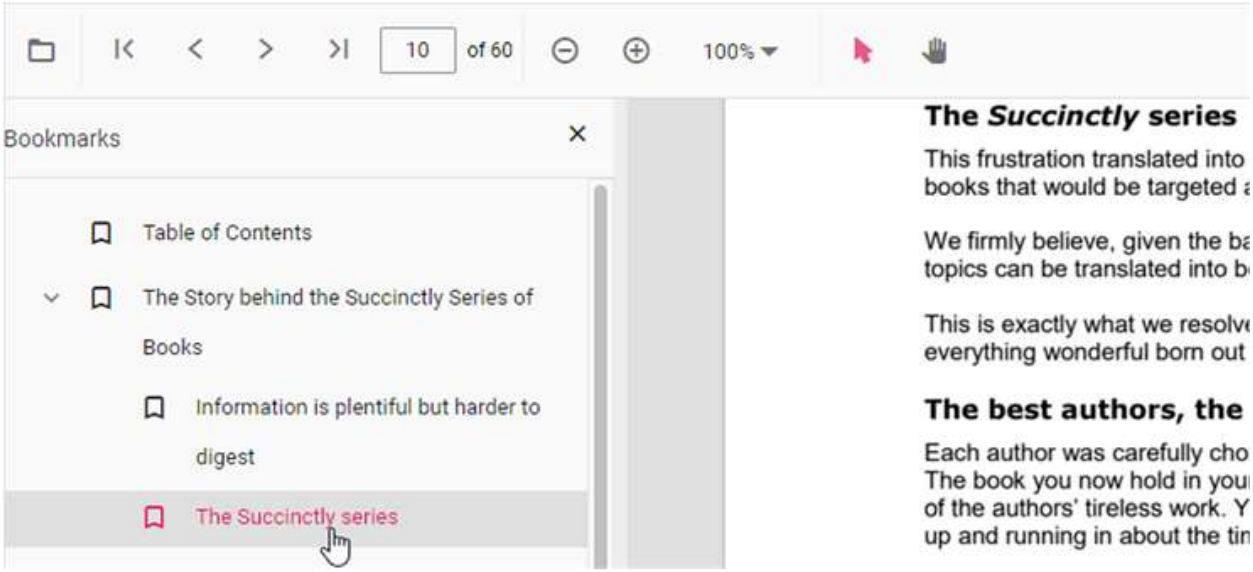
Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:enableBookmark="false">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.



Thumbnail navigation

Thumbnails is the miniature representation of actual pages in PDF files. This feature displays thumbnails of the pages and allows navigation. You can enable/disable thumbnail navigation by using the following code snippet.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:enableTextSearch="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:enableTextSearch="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:enableThumbnail="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
```

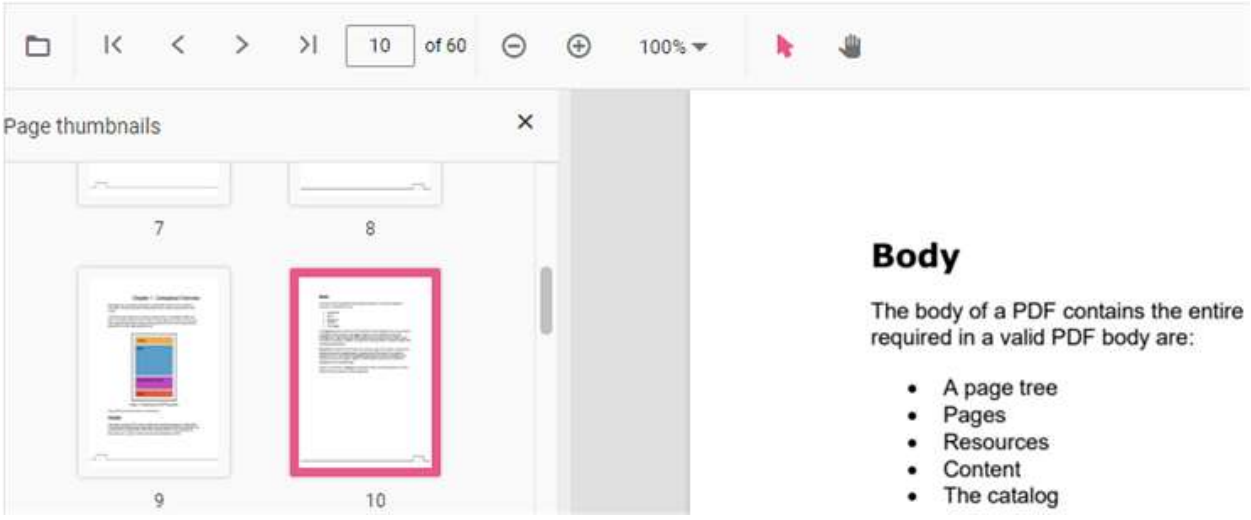
Copyright © 2001 -2024 Syncfusion Inc.

```
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl"
:enableThumbnail="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.



Hyperlink navigation

Hyperlink navigation features enables navigation to the URLs (website links) in a PDF file.

Table of content navigation

Table of contents navigation allows users to navigate to different parts of a PDF file that are listed in the table of contents section.

You can enable/disable link navigation by using the following code snippet.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:enableHyperlink="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl"
:enableHyperlink="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

You can change the open state of the hyperlink in the PDF Viewer by using the following code snippet.

STANDALONE

```
<template>
<div id="app">
```

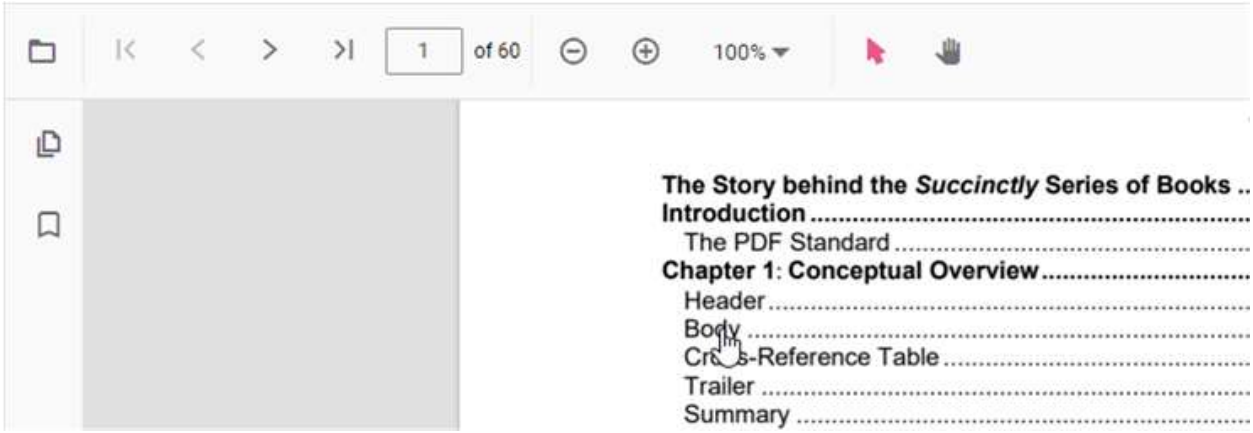
```
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:enableHyperlink="true"
:hyperlinkOpenState="hyperlinkOpenState">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
hyperlinkOpenState:"NewTab"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:enableHyperlink="true"
:hyperlinkOpenState="hyperlinkOpenState">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
hyperlinkOpenState:"NewTab"
};
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```



See also

- Toolbar items
- Feature Modules

Magnification in Vue Pdfviewer component

The magnification tools of the PDF Viewer contains ZoomIn, ZoomOut, Zoom, FitPage, and FitWidth tools in the

default toolbar. The PDF Viewer also has an option to show or hide the magnification tools in the default toolbar.

The following code snippet describes how to enable the magnification in PDF Viewer.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:enableMagnification="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

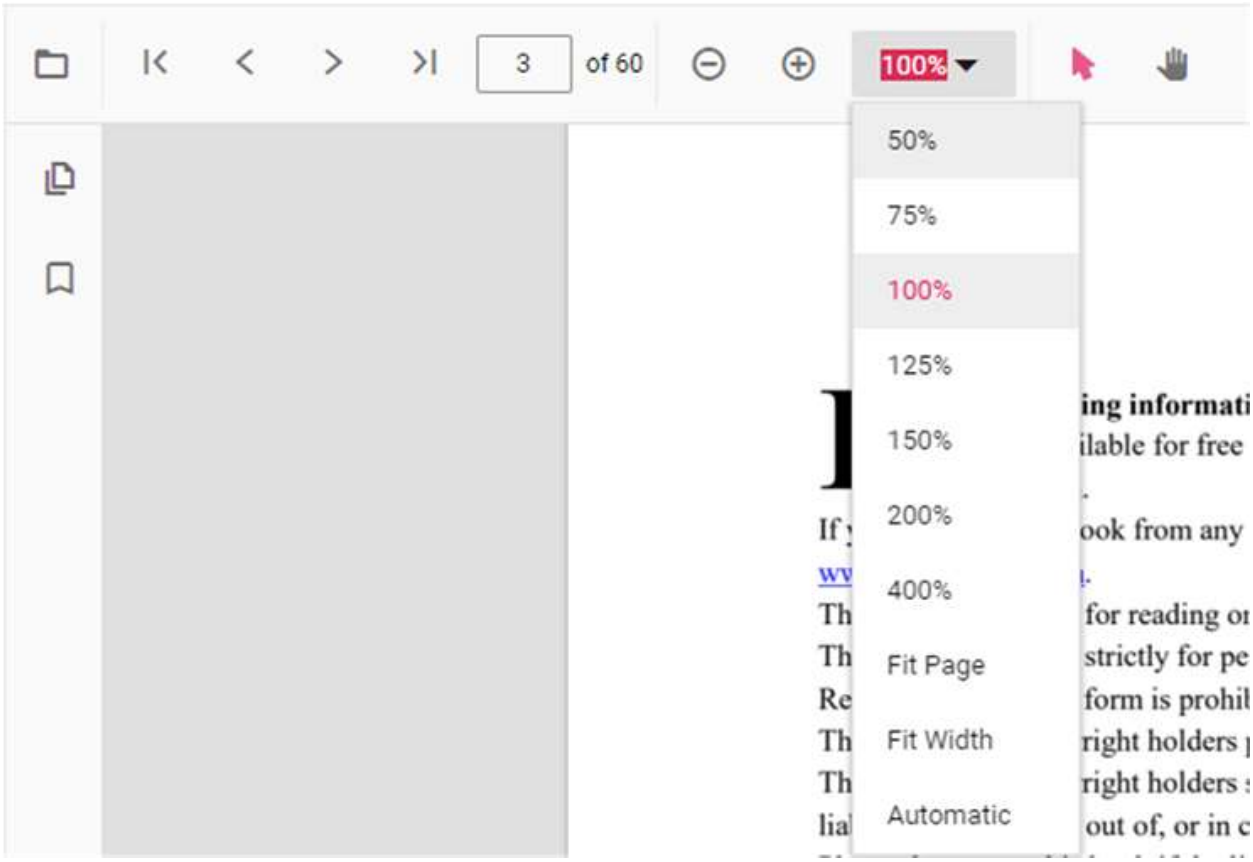
SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl"
:enableMagnification="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

The following magnification options are available in the default toolbar of PDF Viewer,

- **ZoomIn**:- Zoom in from the current zoom value of PDF pages.
- **ZoomOut**:- Zoom out from the current zoom value of PDF pages.
- **Zoom**:- Zoom to specific zoom value of PDF pages.
- **FitPage**:- Fits the page width with in the available view port size.
- **FitWidth**:- Fits the view port width based on the page content size.

Copyright © 2001 -2024 Syncfusion Inc.



PDF Viewer can support the zoom value ranges from 50 to 400.

See also

- Toolbar items
- Feature Modules

Text search in Vue Pdfviewer component

The Text Search option in PDF Viewer is used to find and highlight the text content from the document. You can enable/disable the text search using the following code snippet.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:enableTextSearch="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
```

```
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

SERVER-BACKED

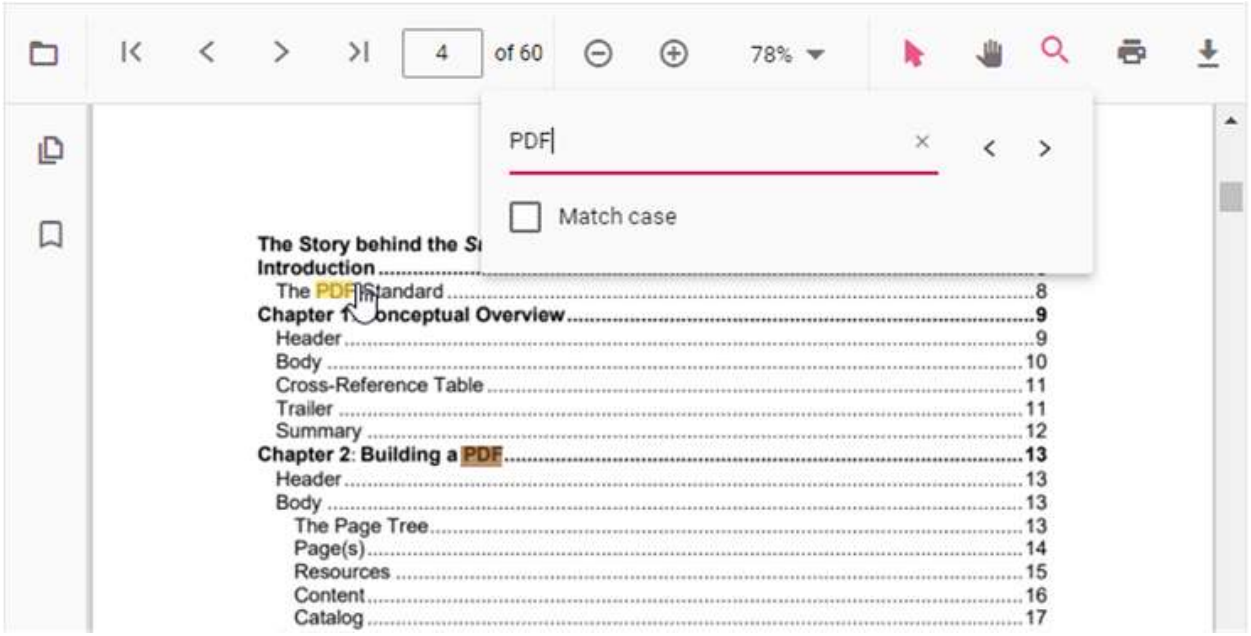
```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:enableTextSearch="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

The following text search methods are available in the PDF Viewer,

- Search text:- Searches the target text in the PDF document and highlights the occurrences in the pages.
- Search next:- Searches the next occurrence of the searched text from the current occurrence of the PdfViewer.

Copyright © 2001 -2024 Syncfusion Inc.

- **Search previous**:- Searches the previous occurrence of the searched text from the current occurrence of the PdfViewer.
- **Cancel text search**:- The text search can be cancelled and the highlighted occurrences from the PDF Viewer can be removed .



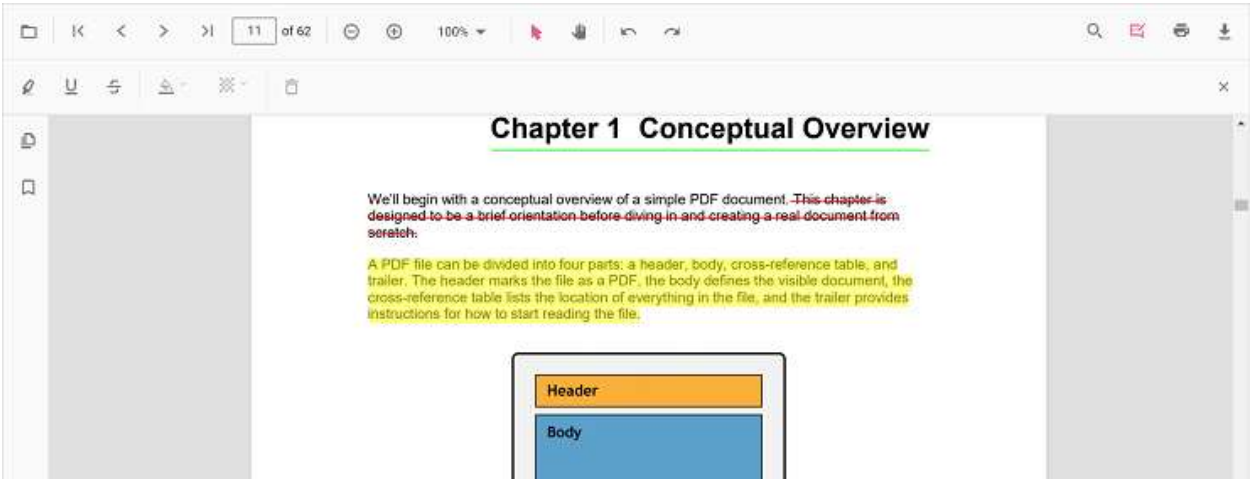
See also

- Toolbar items
- Feature Modules

Annotation

Text markup annotation in Vue Pdfviewer component

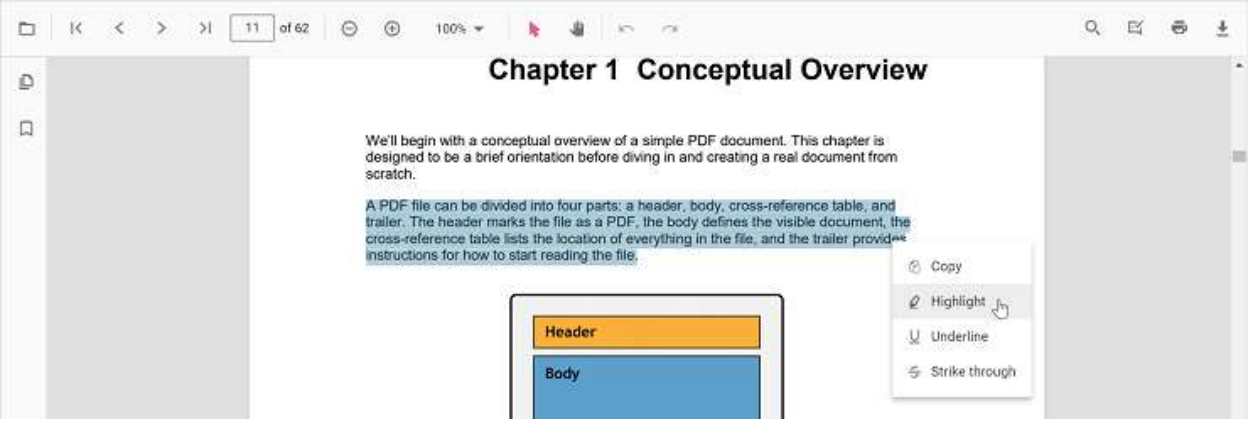
The PDF Viewer control provides the options to add, edit, and delete text markup annotations such as highlight, underline, and strikethrough annotations in the PDF document.



Highlight a text

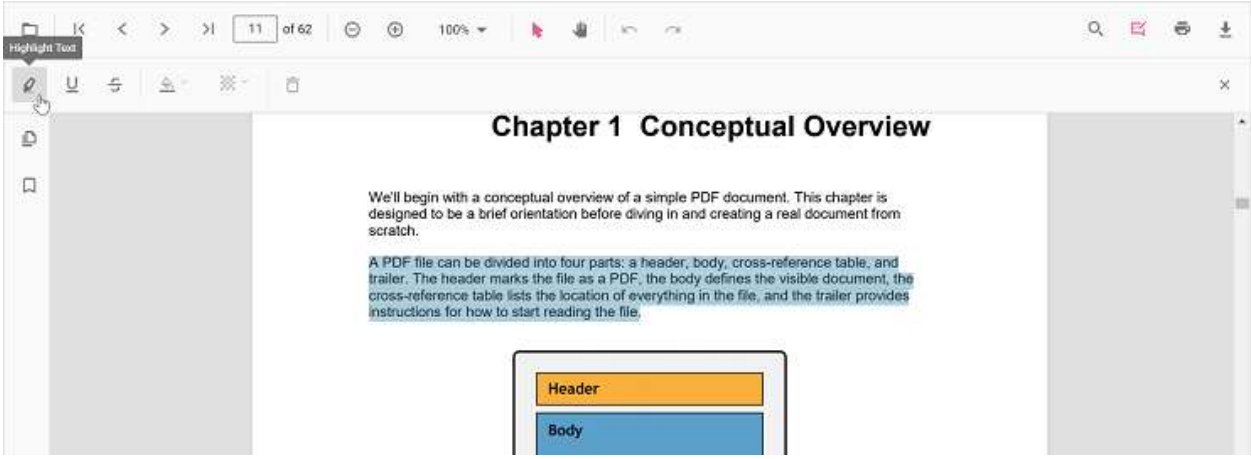
There are two ways to highlight a text in the PDF document:

1.Using the context menu *Select a text in the PDF document and right-click it.* Select **Highlight** option in the context menu that appears.



<!-- markdownlint-disable MD029 -->

2.Using the annotation toolbar *Click the Edit Annotation button in the PDF Viewer toolbar. A toolbar appears below it.* Select the **Highlight** button in the annotation toolbar. It enables the highlight mode. *Select the text and the highlight annotation will be added.* You can also select the text and apply the highlight annotation using the **Highlight** button.



In the pan mode, if the highlight mode is entered, the PDF Viewer control will switch to text select mode to enable the text selection for highlighting the text.

Refer to the following code snippet to switch to highlight mode.

STANDALONE

```
<template>
<div id="app">
<button id="set">Highlight</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
```

```
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Highlight');
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button id="set">Highlight</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Highlight');
});
}
}
}
</script>
```

Refer to the following code snippet to switch back to normal mode from highlight mode.

STANDALONE

```
<template>
<div id="app">
<button id="set">Highlight</button>
<button id="setNone">Normal Mode</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
```

```
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Highlight');
});
document.getElementById('setNone').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('None');
});
}
}
}
</script>
```

SERVER-BACKED

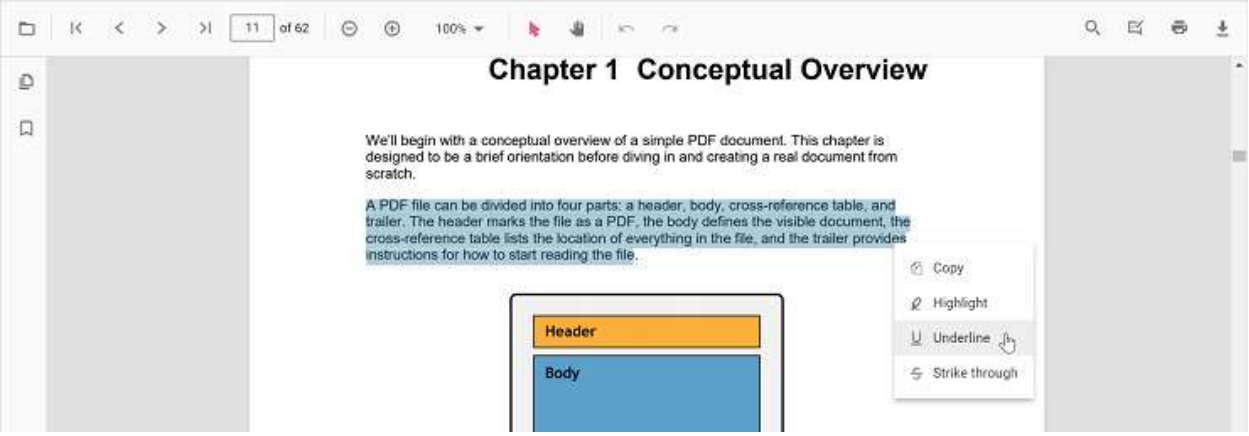
```
<template>
<div id="app">
<button id="set">Highlight</button>
<button id="setNone">Normal Mode</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Highlight');
});
```

```
document.getElementById('setNone').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('None');
});
}
}
}
</script>
```

Underline a text

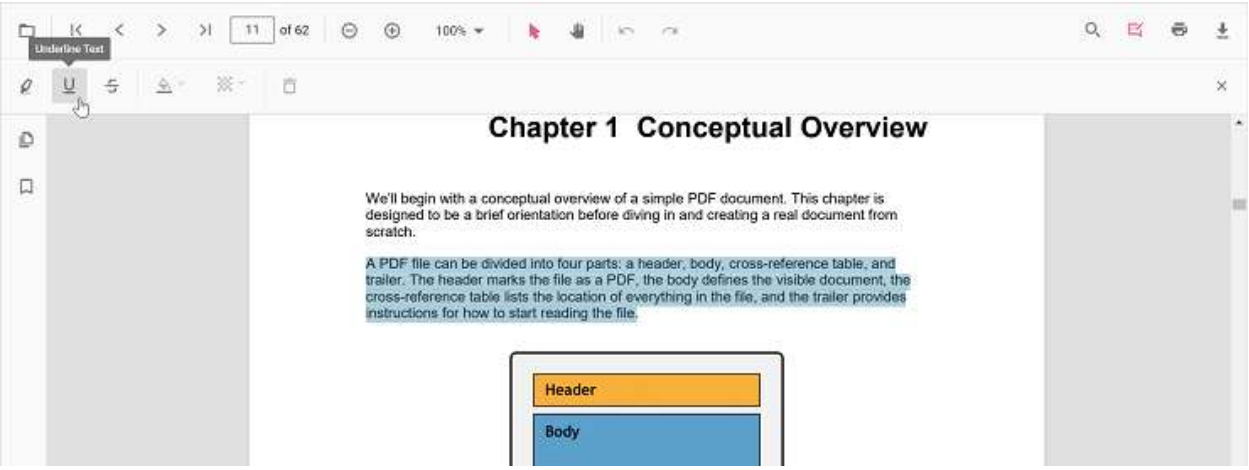
There are two ways to underline a text in the PDF document:

1.Using the context menu *Select a text in the PDF document and right-click it.* Select **Underline** option in the context menu that appears.



<!-- markdownlint-disable MD029 -->

2.Using the annotation toolbar *Click the Edit Annotation button in the PDF Viewer toolbar. A toolbar appears below it.* Select the **Underline** button in the annotation toolbar. It enables the underline mode. *Select the text and the underline annotation will be added.* You can also select the text and apply the underline annotation using the **Underline** button.



In the pan mode, if the underline mode is entered, the PDF Viewer control will switch to text select mode to enable the text selection for underlining the text.

Refer to the following code snippet to switch to underline mode.

Copyright © 2001 -2024 Syncfusion Inc.

STANDALONE

```
<template>
<div id="app">
<button id="set">Underline</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Underline');
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button id="set">Underline</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Underline');
});
}
}
}
</script>
```

Refer to the following code snippet to switch back to normal mode from underline mode.

STANDALONE

```
<template>
<div id="app">
<button id="set">Underline</button>
<button id="setNone">Normal Mode</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Underline');
});
document.getElementById('setNone').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('None');
});
}
}
}
</script>
```

SERVER-BACKED

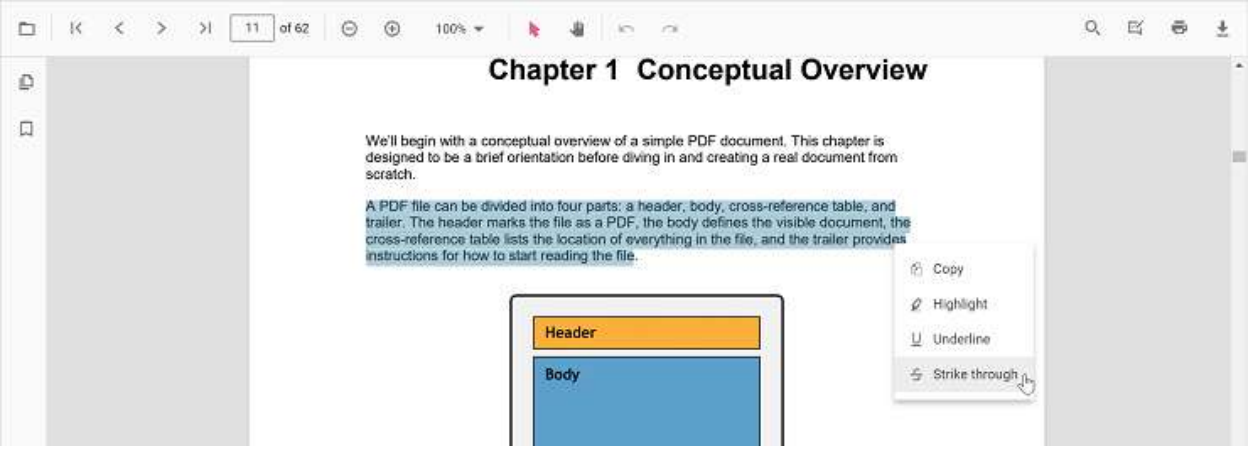
```
<template>
<div id="app">
<button id="set">Underline</button>
<button id="setNone">Normal Mode</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
```

```
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Underline');
});
document.getElementById('setNone').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('None');
});
}
}
}
</script>
```

Strikethrough a text

There are two ways to strikethrough a text in the PDF document:

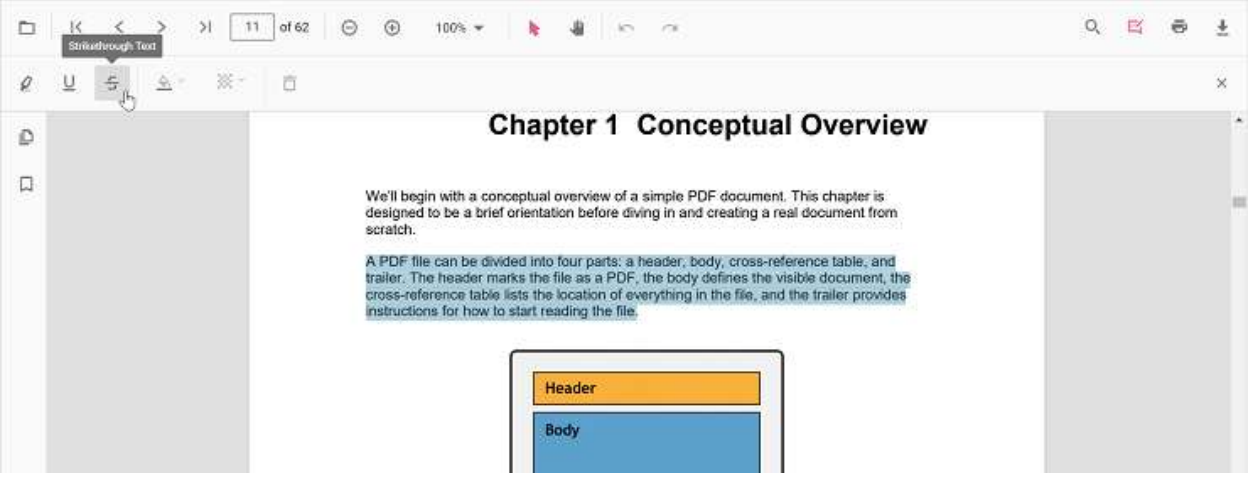
1.Using the context menu *Select a text in the PDF document and right-click it.* Select **Strikethrough** option in the context menu that appears.



<!-- markdownlint-disable MD029 -->

2.Using the annotation toolbar *Click the Edit Annotation button in the PDF Viewer toolbar. A toolbar appears below it.* Select the **Strikethrough** button in the annotation toolbar. It enables the strikethrough mode. *Select the text and the strikethrough annotation will be added.* You can also select the text and apply the strikethrough annotation using the **Strikethrough** button.

Copyright © 2001 -2024 Syncfusion Inc.



In the pan mode, if the strikethrough mode is entered, the PDF Viewer control will switch to text select mode to enable the text selection for striking through the text.

Refer to the following code snippet to switch to strikethrough mode.

STANDALONE

```
<template>
<div id="app">
<button id="set">Strikethrough</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
```

```
viewer.annotation.setAnnotationMode('Strikethrough');
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button id="set">Strikethrough</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Strikethrough');
});
}
}
}
</script>
```

Refer to the following code snippet to switch back to normal mode from strikethrough mode.

STANDALONE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<div id="app">
<button id="set">Strikethrough</button>
<button id="setNone">Normal Mode</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Strikethrough');
});
document.getElementById('setNone').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('None');
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button id="set">Strikethrough</button>
<button id="setNone">Normal Mode</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
```

Copyright © 2001 -2024 Syncfusion Inc.

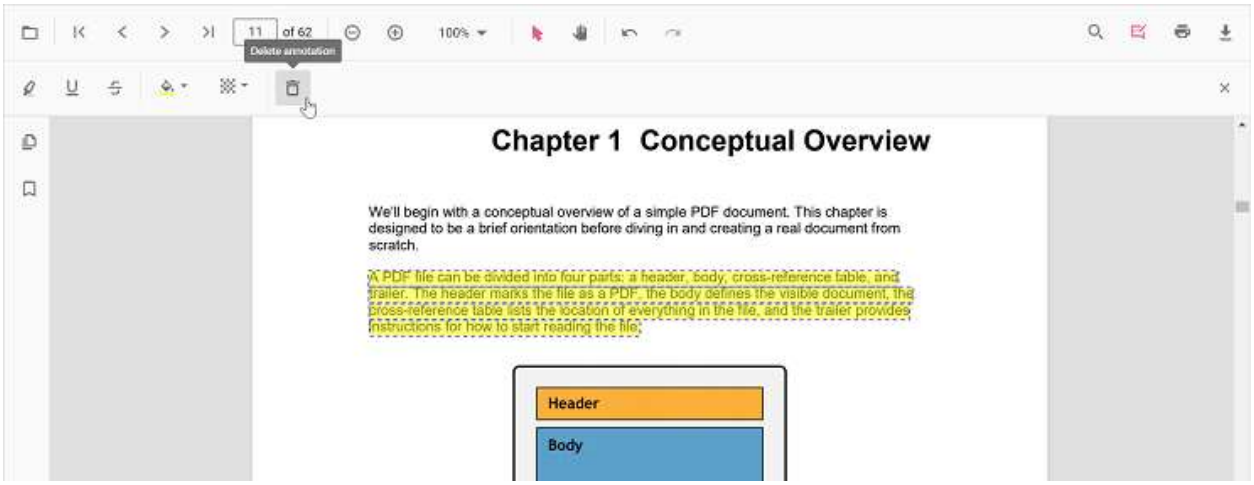
```
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Strikethrough');
});
document.getElementById('setNone').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('None');
});
}
}
}
</script>
```

Deleting a text markup annotation

The selected annotation can be deleted by the following ways:

1.Using Delete key *Select the annotation to be deleted.* Click the Delete key in the keyboard. The selected annotation will be deleted. 2.Using the annotation toolbar *Select the annotation to be deleted.* Click the **Delete Annotation** button in the annotation toolbar. The selected annotation will be deleted.

Copyright © 2001 -2024 Syncfusion Inc.

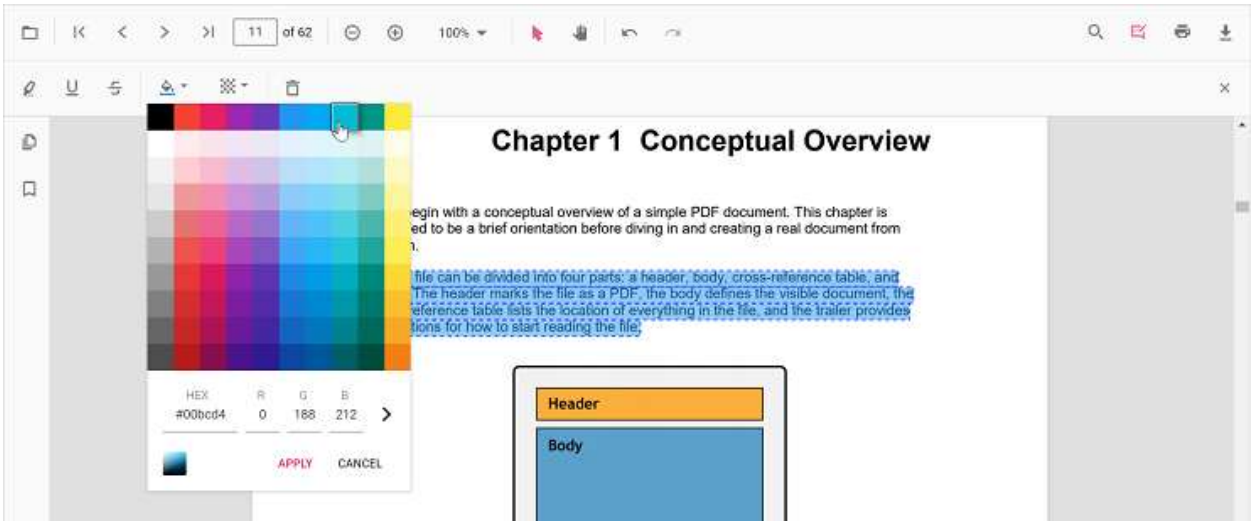


Editing the properties of the text markup annotation

The color and the opacity of the text markup annotation can be edited using the Edit Color tool and the Edit Opacity tool in the annotation toolbar.

Editing color

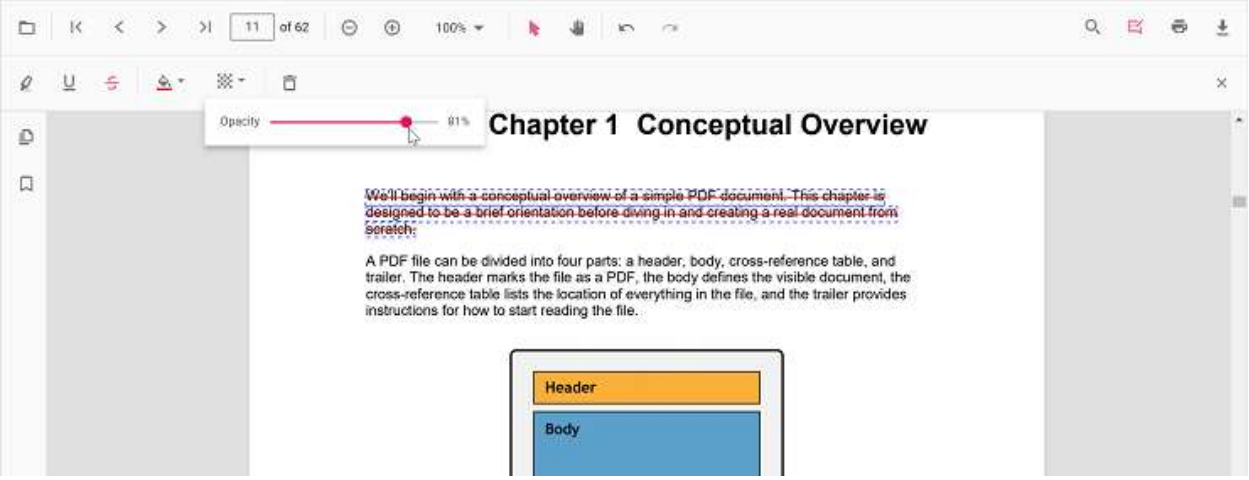
The color of the annotation can be edited using the color palette provided in the Edit Color tool.



Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.

Copyright © 2001 -2024 Syncfusion Inc.



Setting default properties during control initialization

The properties of the text markup annotation can be set before creating the control using highlightSettings, underlineSettings, and strikethroughSettings.

After editing the default color and opacity using the Edit Color tool and Edit Opacity tool, they will be changed to the selected values.

Refer to the following code snippet to set the default annotation settings.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:highlightSettings="highlightSettings"
:underlineSettings="underlineSettings"
:strikethroughSettings="strikethroughSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
highlightSettings: {author: 'Guest User', subject: 'Important', color:
'#ffff00', opacity: 0.9, modifiedDate: ''},
underlineSettings: {author: 'Guest User', subject: 'Points to be
remembered', color: '#00ffff', opacity: 0.9, modifiedDate: ''},
strikethroughSettings: {author: 'Guest User', subject: 'Not Important',
color: '#ff00ff', opacity: 0.9, modifiedDate: ''}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:highlightSettings="highlightSettings"
:underlineSettings="underlineSettings"
:strikethroughSettings="strikethroughSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
highlightSettings: {author: 'Guest User', subject: 'Important', color:
'#ffff00', opacity: 0.9, modifiedDate: ''},
underlineSettings: {author: 'Guest User', subject: 'Points to be
remembered', color: '#00ffff', opacity: 0.9, modifiedDate: ''},
strikethroughSettings: {author: 'Guest User', subject: 'Not Important',
color: '#ff00ff', opacity: 0.9, modifiedDate: ''}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
}
</script>
```

Performing undo and redo

The PDF Viewer performs undo and redo for the changes made in the PDF document. In text markup annotation, undo and redo actions are provided for:

Copyright © 2001 -2024 Syncfusion Inc.

- Inclusion of the text markup annotations.
- Deletion of the text markup annotations.
- Change of either color or opacity of the text markup annotations.

Undo and redo actions can be done by the following ways:

1.Using keyboard shortcuts:

After performing a text markup annotation action, you can undo it by using Ctrl + Z shortcut and redo by using Ctrl + Y shortcut.

2.Using toolbar:

Undo and redo can be done using the **Undo** tool and **Redo** tool provided in the toolbar.

Refer to the following code snippet for calling undo and redo actions from the client-side.

STANDALONE

```
<template>
<div id="app">
<button id="undo">Undo</button>
<button id="redo">Redo</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('undo').addEventListener('click', ()=> {
viewer.undo();
});
document.getElementById('redo').addEventListener('click', ()=> {
viewer.redo();
```

```
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button id="undo">Undo</button>
<button id="redo">Redo</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('undo').addEventListener('click', ()=> {
viewer.undo();
});
document.getElementById('redo').addEventListener('click', ()=> {
viewer.redo();
});
}
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

Saving the text markup annotation

When you click the download tool in the toolbar, the text markup annotations will be saved in the PDF document. This action will not affect the original document.

Printing the text markup annotation

When the print tool is selected in the toolbar, the PDF document will be printed along with the text markup annotations added to the pages. This action will not affect the original document.

Disabling text markup annotation

The PDF Viewer control provides an option to disable the text markup annotation feature. The code snippet for disabling the feature is as follows.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:enableTextMarkupAnnotation="false">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl"
:enableTextMarkupAnnotation="false">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
}
</script>
```

See also

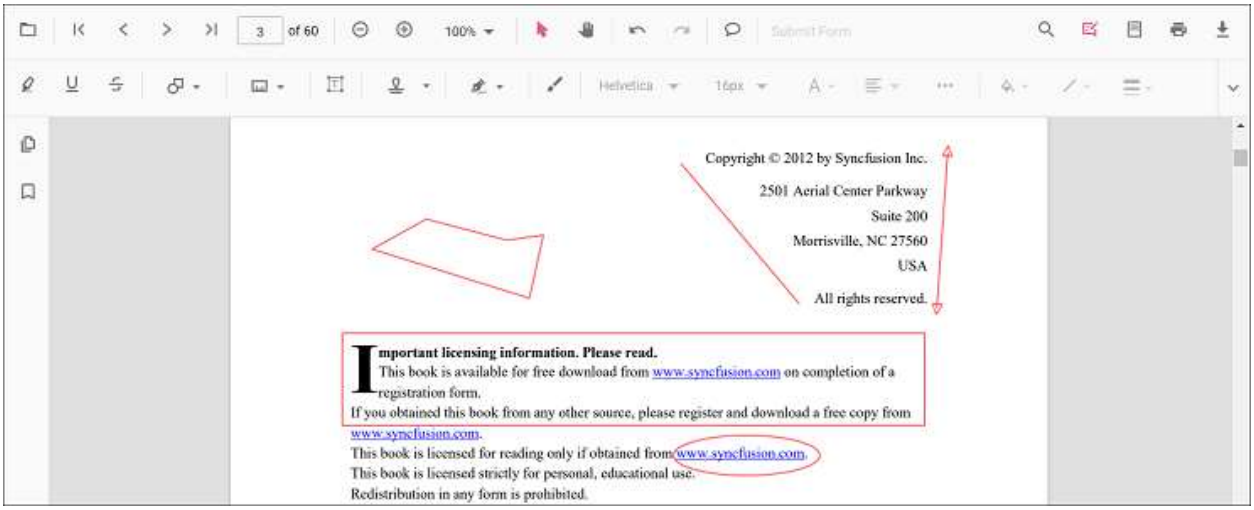
- Toolbar items
- Feature Modules

Shape annotation in Vue Pdfviewer component

The PDF Viewer control provides the options to add, edit, and delete the shape annotations. The shape annotation types supported in the PDF Viewer control are:

- Line
- Arrow
- Rectangle
- Circle
- Polygon

Copyright © 2001 -2024 Syncfusion Inc.

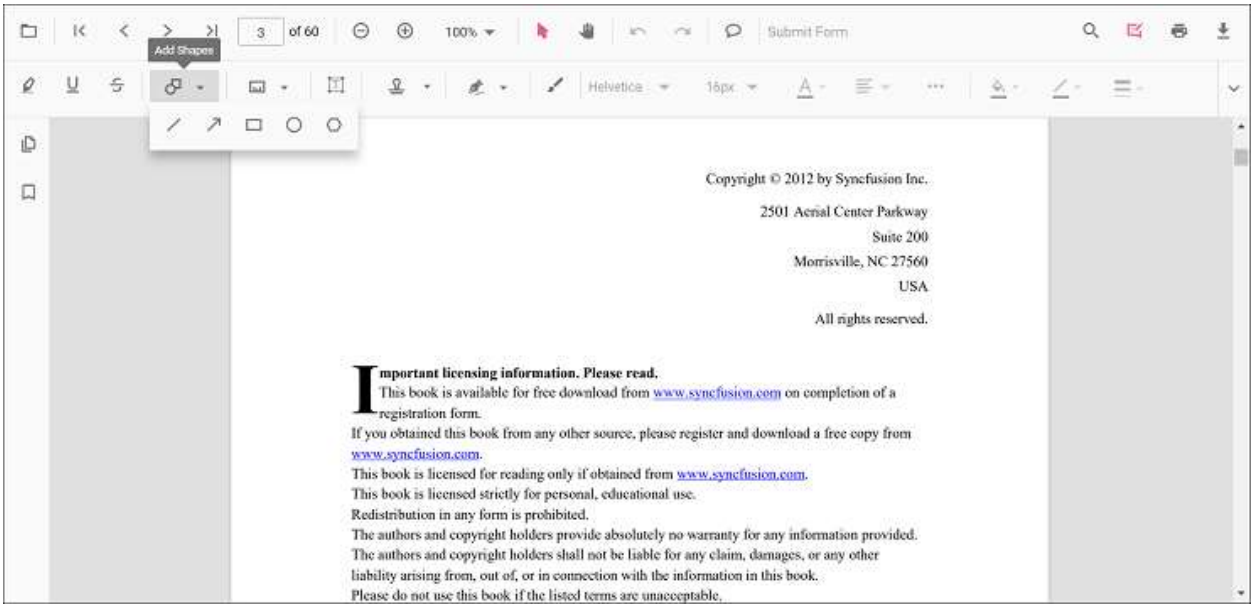


Adding a shape annotation to the PDF document

Shape annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Shape Annotation** drop-down button. A drop-down pop-up will appear and shows the shape annotations to be added.
- Select the shape types to be added to the page in the drop-down pop-up. It enables the selected shape annotation mode.
- You can add the shapes over the pages of the PDF document.

In the pan mode, if the shape annotation mode is entered, the PDF Viewer control will switch to text select mode.



Refer to the following code sample to switch to the circle annotation mode.

STANDALONE

```
<template>
```

```
<div id="app">
<button id="set">Circle</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Circle');
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button id="set">Circle</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

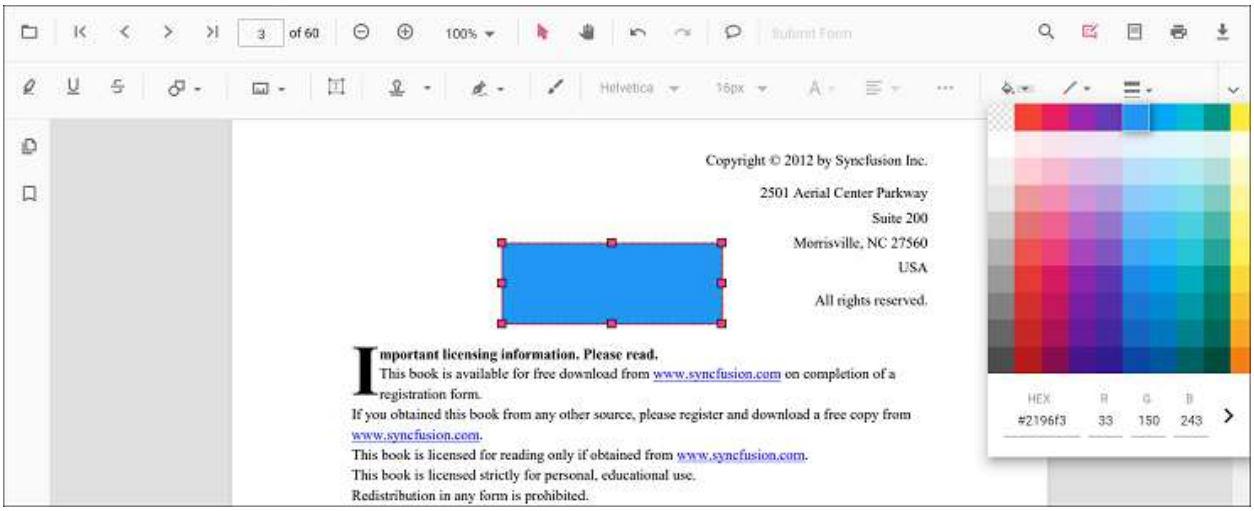
```
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Circle');
});
}
}
}
</script>
```

Editing the properties of the shape annotation

The fill color, stroke color, thickness, and opacity of the shape annotation can be edited using the Edit color tool, Edit stroke color tool, Edit thickness tool, and Edit opacity tool in the annotation toolbar.

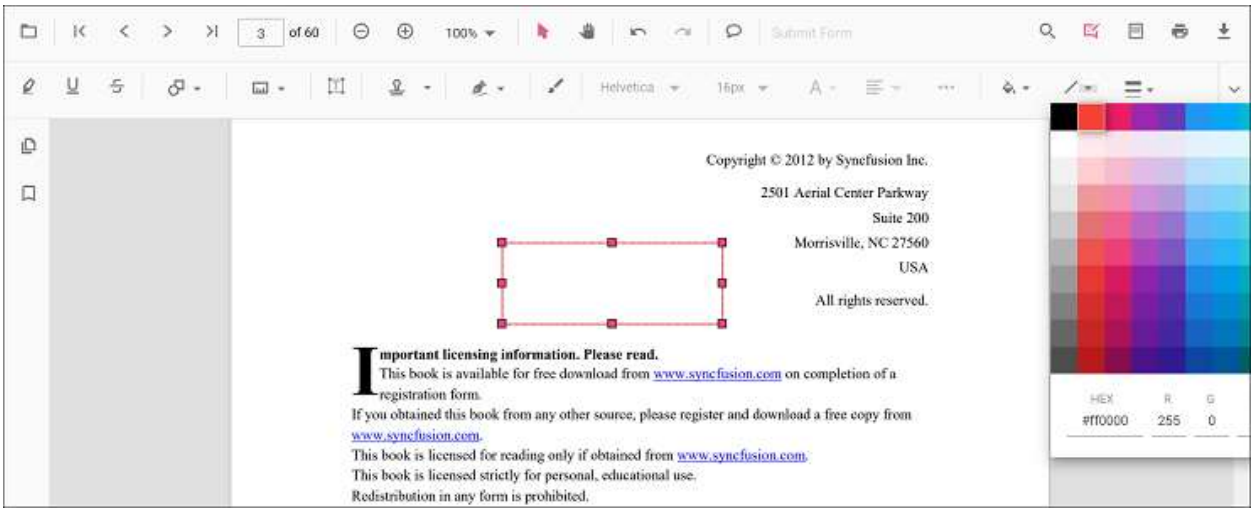
Editing fill color

The fill color of the annotation can be edited using the color palette provided in the Edit Color tool.



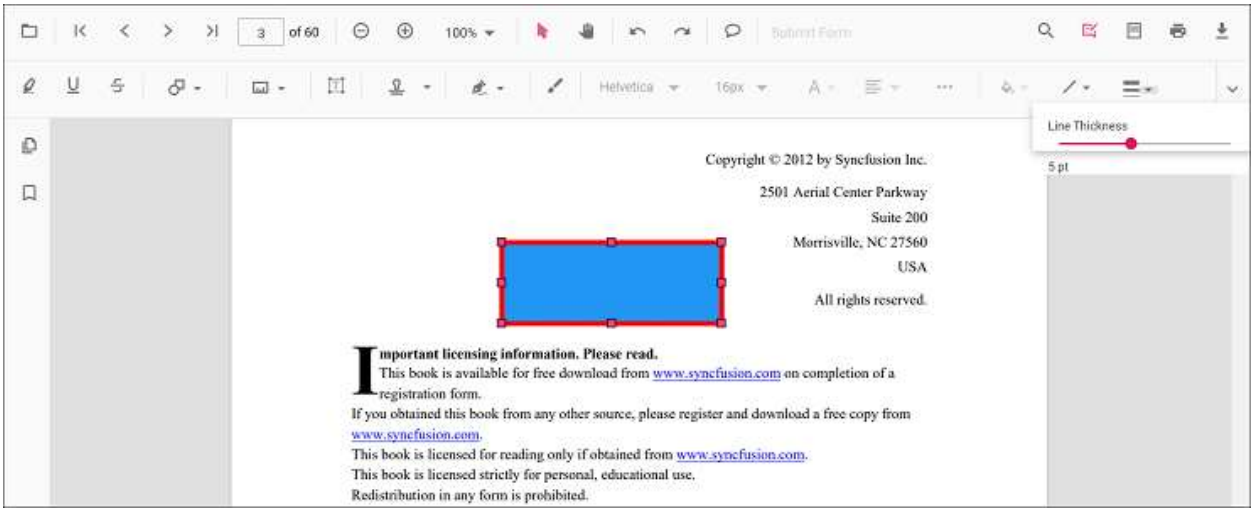
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



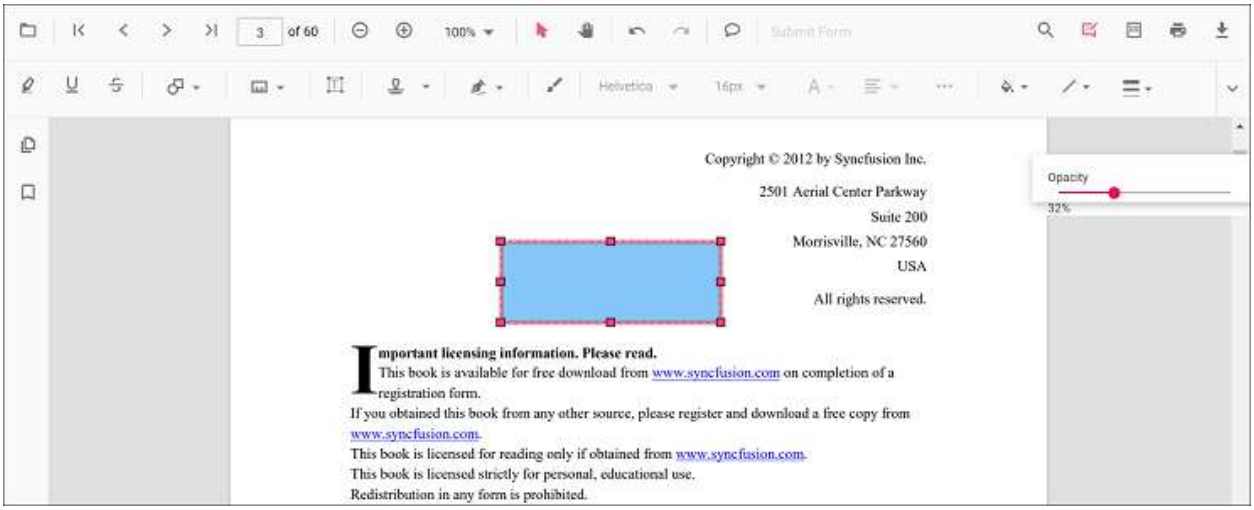
Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit Thickness tool.



Editing opacity

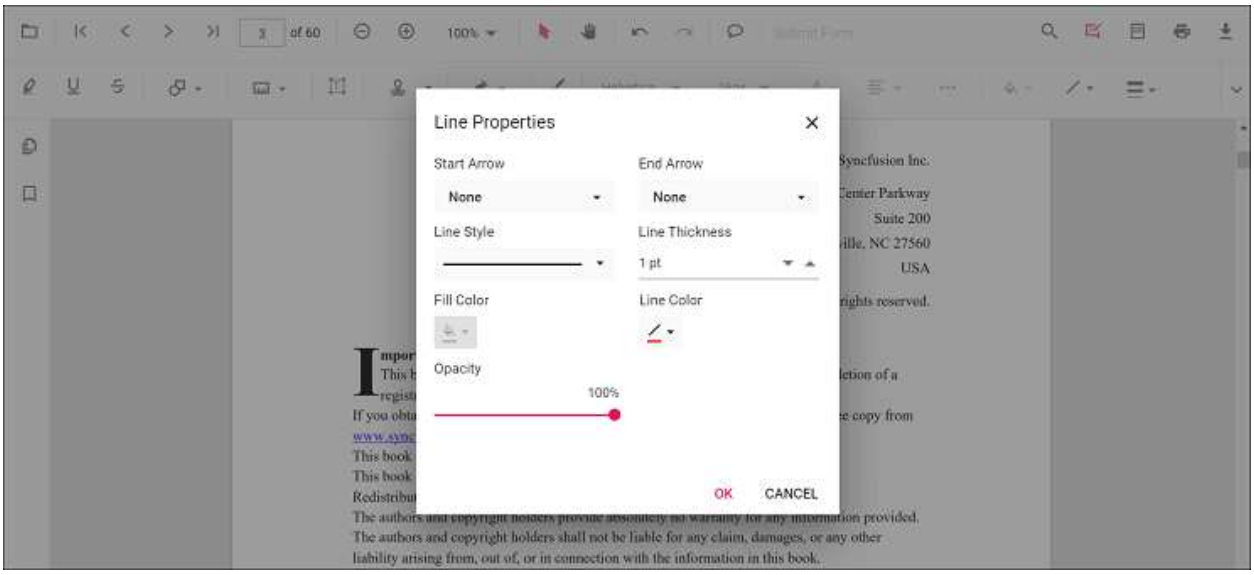
The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Editing the line properties

The properties of the line shapes such as line and arrow annotations can be edited using the Line Properties window. It can be opened by selecting the Properties option in the context menu that appears on right-clicking the line and arrow annotations.

Refer to the following code sample to set the default annotation settings.



Setting default properties during the control initialization

The properties of the shape annotations can be set before creating the control using LineSettings, ArrowSettings, RectangleSettings, CircleSettings, and PolygonSettings.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:lineSettings="lineSettings"
:arrowSettings="arrowSettings"
```

```
:rectangleSettings="rectangleSettings"
:circleSettings="circleSettings"
:polygonSettings="polygonSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
lineSettings: {fillColor: 'blue', opacity: 0.6, strokeColor: 'green'},
arrowSettings: {fillColor: 'green', opacity: 0.6, strokeColor: 'blue'},
rectangleSettings: {fillColor: 'yellow', opacity: 0.6, strokeColor:
'orange'},
circleSettings: {fillColor: 'orange', opacity: 0.6, strokeColor: 'pink'},
polygonSettings: {fillColor: 'pink', opacity: 0.6, strokeColor: 'yellow'}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:lineSettings="lineSettings"
:arrowSettings="arrowSettings"
:rectangleSettings="rectangleSettings"
:circleSettings="circleSettings"
:polygonSettings="polygonSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
```

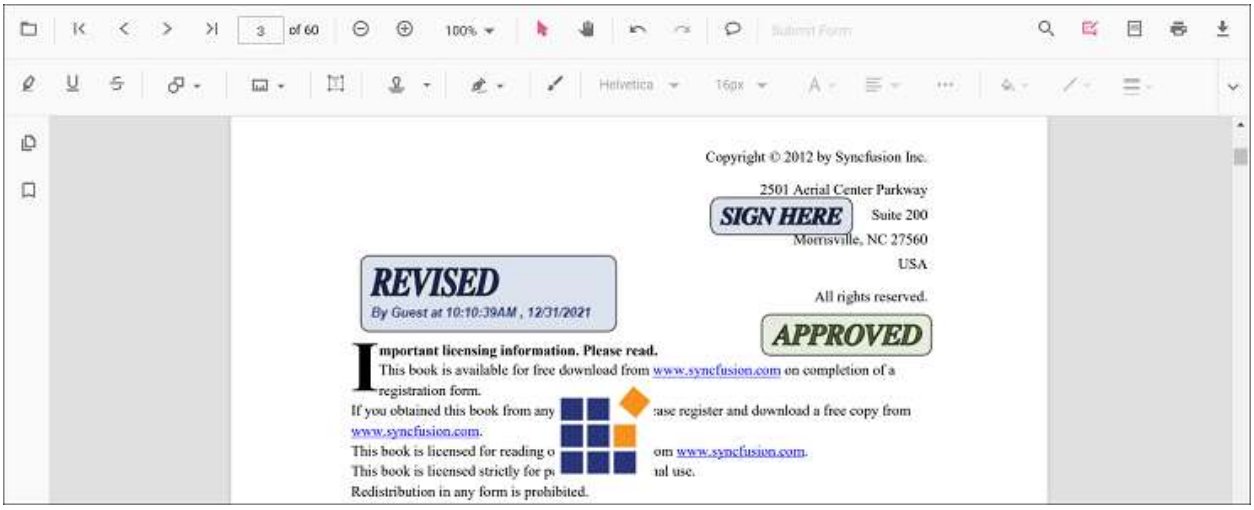
Copyright © 2001 -2024 Syncfusion Inc.

```
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
lineSettings: {fillColor: 'blue', opacity: 0.6, strokeColor: 'green'},
arrowSettings: {fillColor: 'green', opacity: 0.6, strokeColor: 'blue'},
rectangleSettings: {fillColor: 'yellow', opacity: 0.6, strokeColor:
'orange'},
circleSettings: {fillColor: 'orange', opacity: 0.6, strokeColor: 'pink'},
polygonSettings: {fillColor: 'pink', opacity: 0.6, strokeColor: 'yellow'}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
}
</script>
```

Stamp annotation in Vue Pdfviewer component

The PDF Viewer control provides options to add, edit, delete, and rotate the following stamp annotation in the PDF documents:

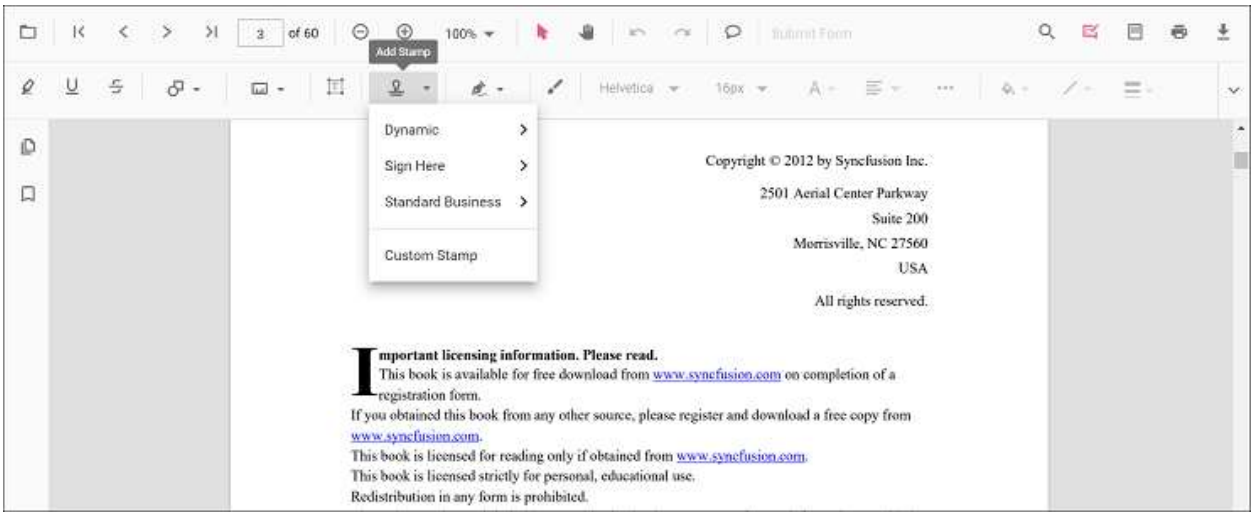
- Dynamic
- Sign Here
- Standard Business
- Custom Stamp



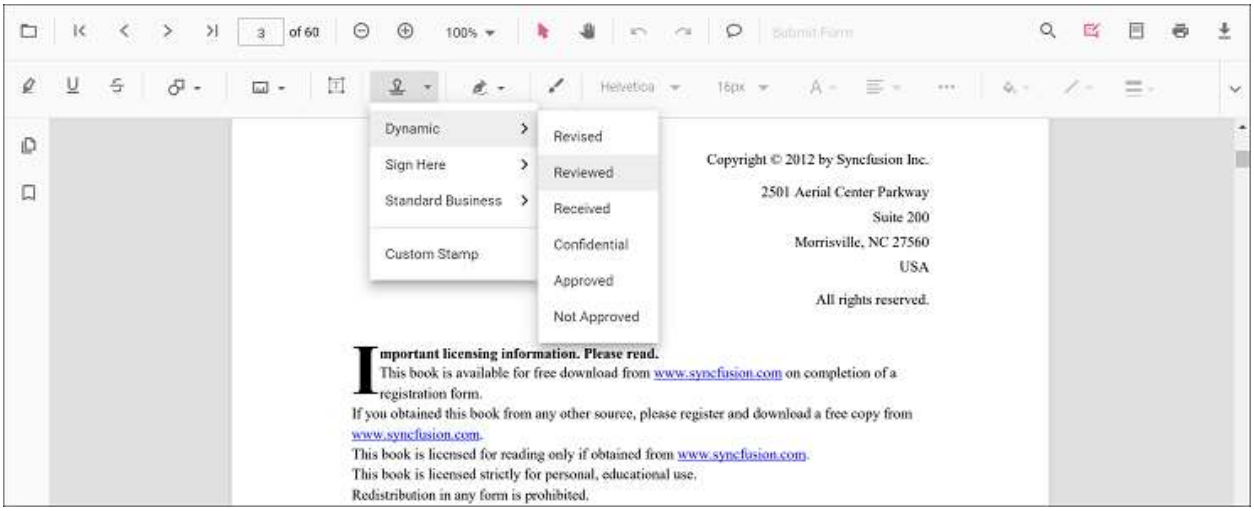
Adding stamp annotations to the PDF document

The stamp annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Stamp Annotation** drop-down button. A drop-down pop-up will appear and shows the stamp annotations to be added.



- Select the annotation type to be added to the page in the pop-up.



- You can add the annotation over the pages of the PDF document.

In the pan mode, if the stamp annotation mode is entered, the PDF Viewer control will switch to text select mode.

Refer to the following code sample to switch to the stamp annotation mode.

STANDALONE

```
<template>
<div id="app">
<button id="set">Stamp</button>
<ejs-pdfviewer
id="pdfViewer"
```

```
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print, SignStampItem,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Stamp', null, SignStampItem.Witness);
});
}
}
}
</script>
```

SERVER-BACKED

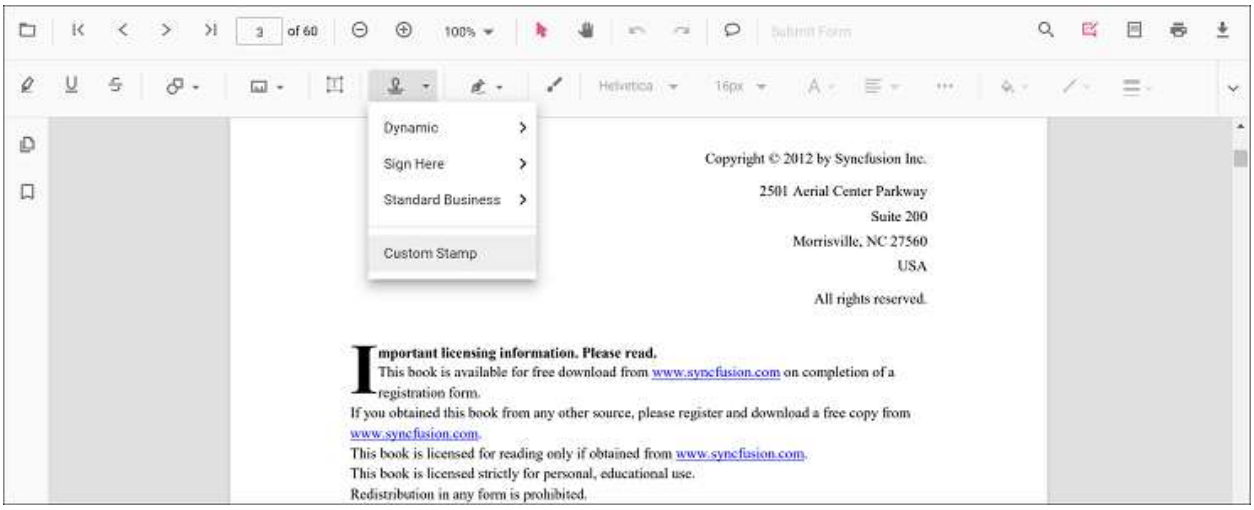
```
<template>
<div id="app">
<button id="set">Stamp</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print, SignStampItem,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Stamp', null, SignStampItem.Witness);
});
}
}
}
</script>
```

Adding custom stamp to the PDF document

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Stamp Annotation** drop-down button. A drop-down pop-up will appear and shows the stamp annotations to be added.
- Click the Custom Stamp button.



- The file explorer dialog will appear, choose the image and then add the image to the PDF page.

The JPG and JPEG image format is only supported in the custom stamp annotations.

Setting default properties during control initialization

The properties of the stamp annotation can be set before creating the control using the StampSettings.

After editing the default opacity using the Edit Opacity tool, they will be changed to the selected values.

Refer to the following code sample to set the default sticky note annotation settings.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:stampSettings="stampSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
stampSettings: {opacity: 0.3, author: 'Guest User'}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
}
</script>
```

SERVER-BACKED

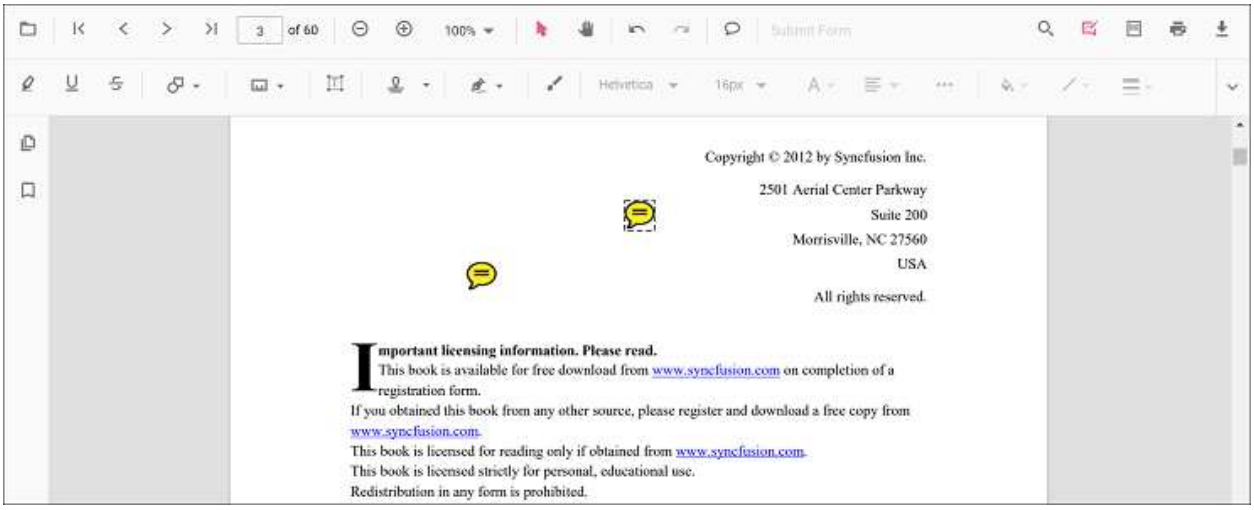
```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:stampSettings="stampSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
stampSettings: {opacity: 0.3, author: 'Guest User'}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
}
</script>
```

Sticky notes annotation in Vue Pdfviewer component

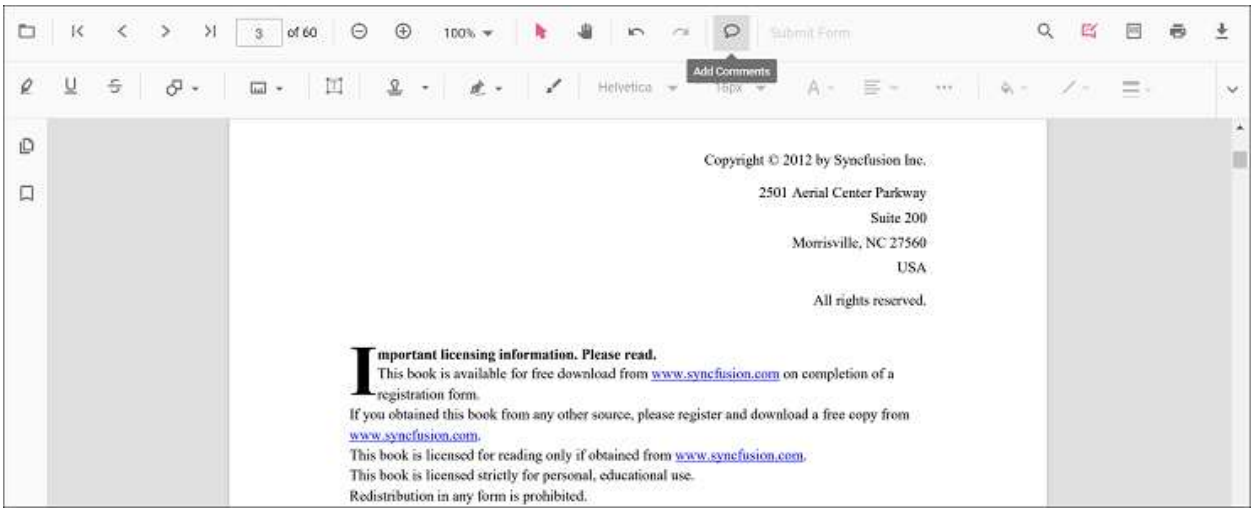
The PDF Viewer control provides the options to add, edit, and delete the sticky note annotations in the PDF document.



Adding a sticky note annotation to the PDF document

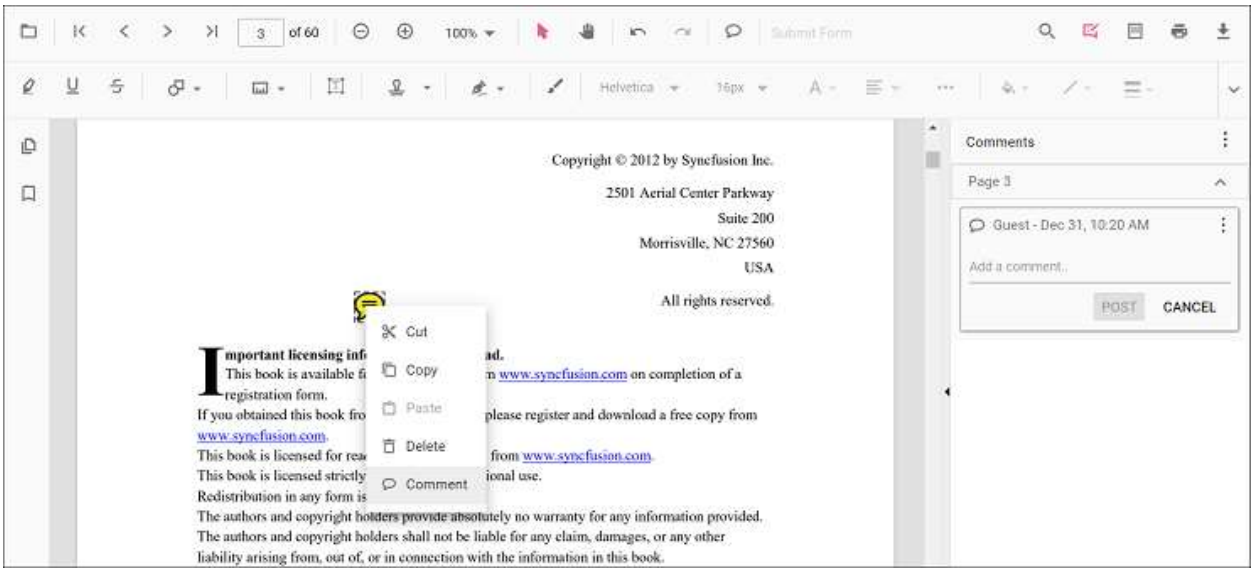
Sticky note annotations can be added to the PDF document using the annotation toolbar.

- Click the **Comments** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the position where you want to add sticky note annotation in the PDF document.
- Sticky note annotation will be added in the clicked positions.



Annotation comments can be added to the PDF document using the comment panel.

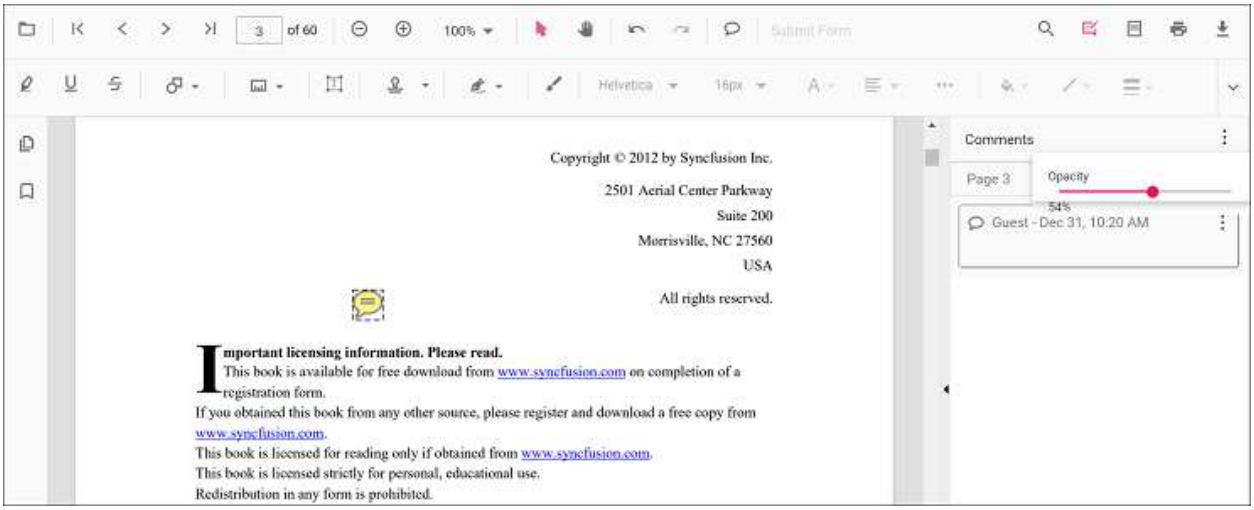
- Select a Sticky note annotation in the PDF document and right-click it.
- Select the Comment option in the context menu that appears.
- Now, you can add Comments, Reply, and Status using the Comment Panel.
- Now, you can add Comments, Reply, and Status using the Comment Panel.



Editing the properties of the sticky note annotation

Editing opacity

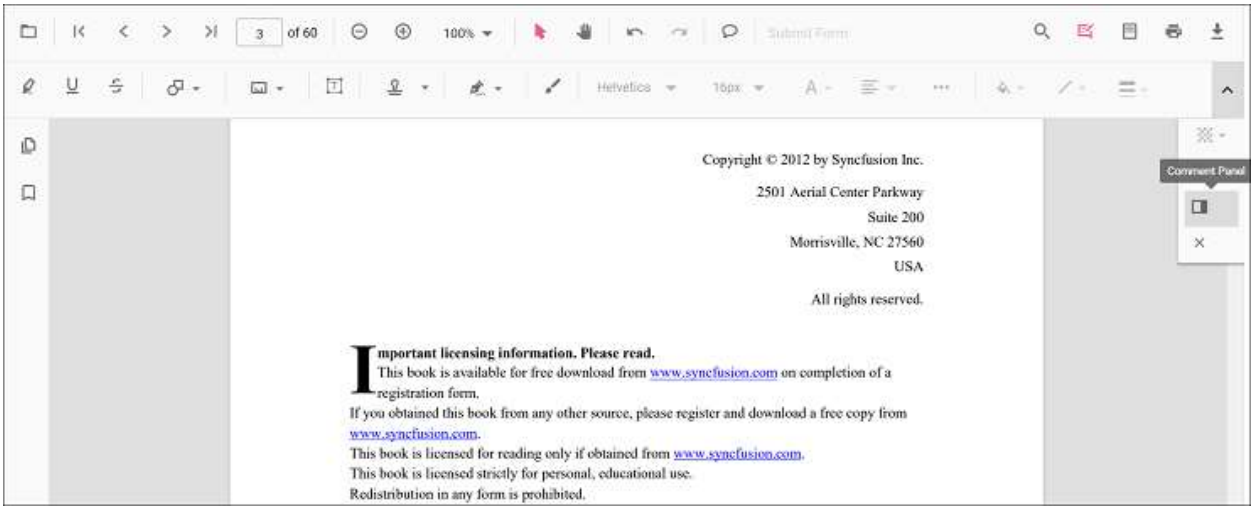
The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



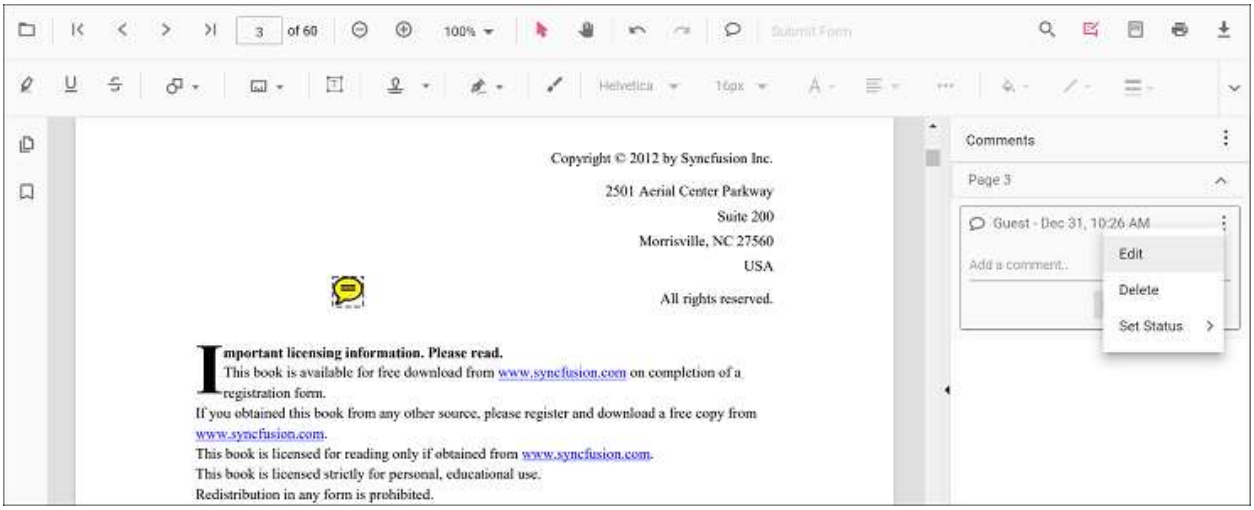
Editing comments

The comment, comment reply, and comment status of the annotation can be edited using the Comment Panel.

- Open the comment panel using the Comment Panel button showing in the annotation toolbar.



You can modify or delete the comments or comments replay and it’s status using the menu option provided in the comment panel.



Setting default properties during the control initialization

The properties of the sticky note annotation can be set before creating the control using the StickyNoteSettings.

After editing the default opacity using the Edit Opacity tool, they will be changed to the selected values. Refer to the following code sample to set the default sticky note annotation settings.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:stickyNotesSettings="stickyNotesSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
stickyNotesSettings: {author: 'Syncfusion'}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:stickyNotesSettings="stickyNotesSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
stickyNotesSettings: {author: 'Syncfusion'}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
}
</script>
```

Disabling sticky note annotations

The PDF Viewer control provides an option to disable the sticky note annotations feature. The code sample for disabling the feature is as follows.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:enableStickyNotesAnnotation="enableStickyNotesAnnotation">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
enableStickyNotesAnnotation: false
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
}
</script>
```

SERVER-BACKED

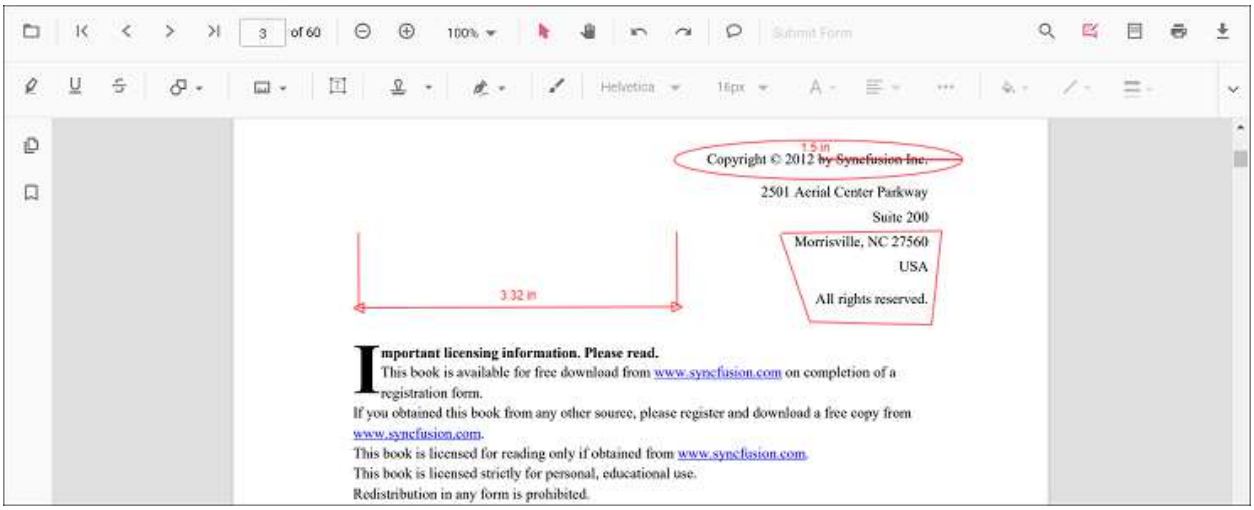
```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:enableStickyNotesAnnotation="enableStickyNotesAnnotation">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
enableStickyNotesAnnotation: false
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

Measurement annotation in Vue Pdfviewer component

The PDF Viewer provides the options to add measurement annotations. You can measure the page annotations with the help of measurement annotation. The supported measurement annotations in the PDF Viewer control are:

- Distance
- Perimeter
- Area
- Radius
- Volume



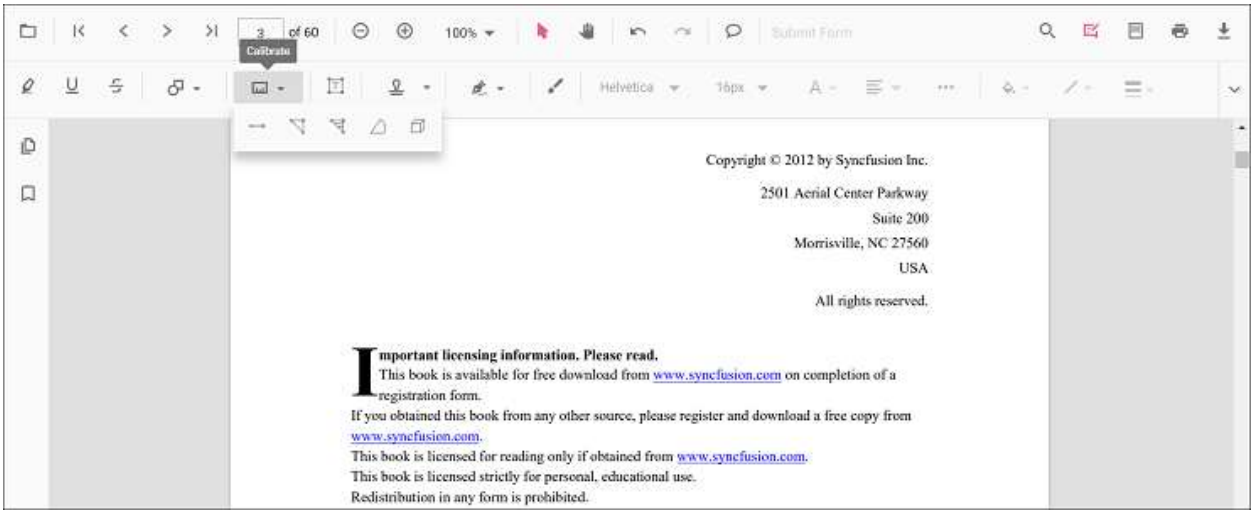
Adding measurement annotations to the PDF document

The measurement annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Measurement Annotation** dropdown button. A dropdown pop-up will appear and shows the measurement annotations to be added.
- Select the measurement type to be added to the page in the dropdown pop-up. It enables the selected measurement annotation mode.
- You can measure and add the annotation over the pages of the PDF document.

In the pan mode, if the measurement annotation mode is entered, the PDF Viewer control will switch to text select mode.

Copyright © 2001 -2024 Syncfusion Inc.



Refer to the following code snippet to switch to distance annotation mode.

STANDALONE

```
<template>
<div id="app">
<button id="set">Distance</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Distance');
});
```

```
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button id="set">Distance</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Distance');
});
}
}
}
</script>
```

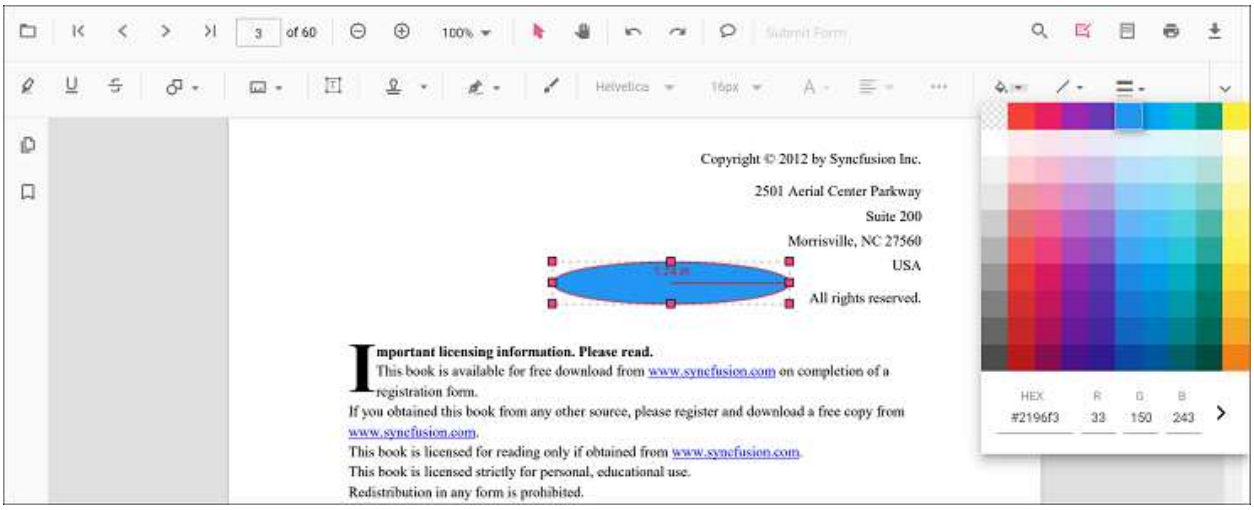
Editing the properties of measurement annotation

The fill color, stroke color, thickness, and opacity of the measurement annotation can be edited using the Edit Color tool, Edit Stroke Color tool, Edit Thickness tool, and Edit Opacity tool in the annotation toolbar.

Copyright © 2001 -2024 Syncfusion Inc.

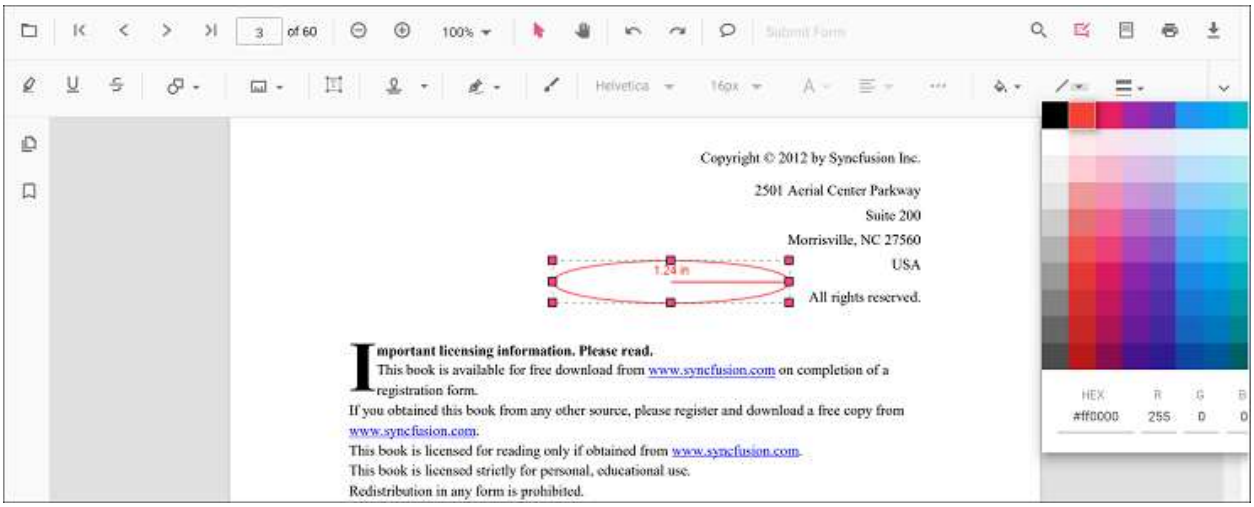
Editing fill color

The fill color of the annotation can be edited using the color palette provided in the Edit Color tool.



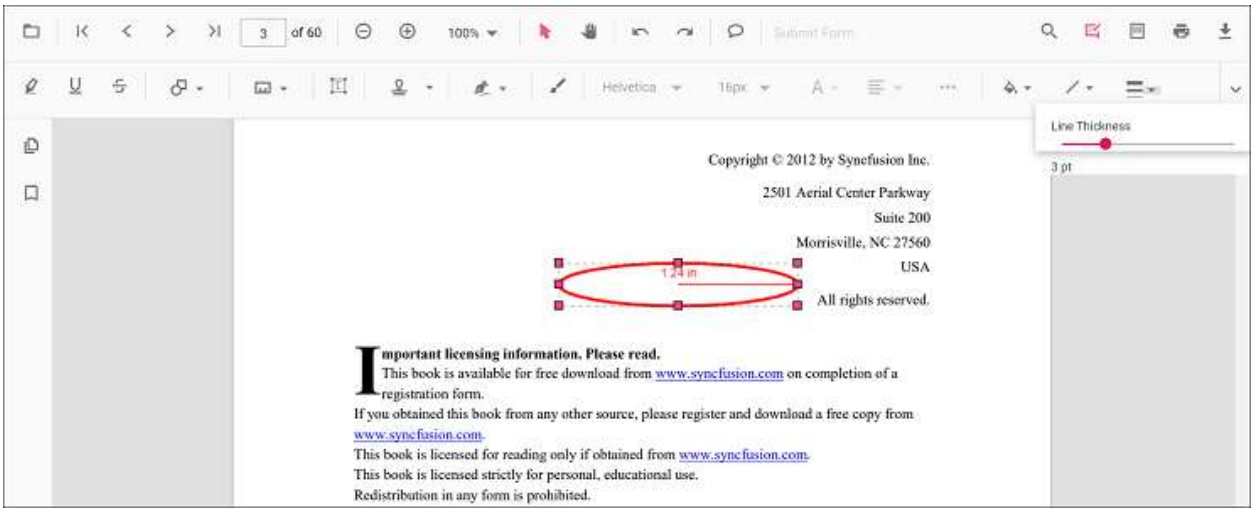
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



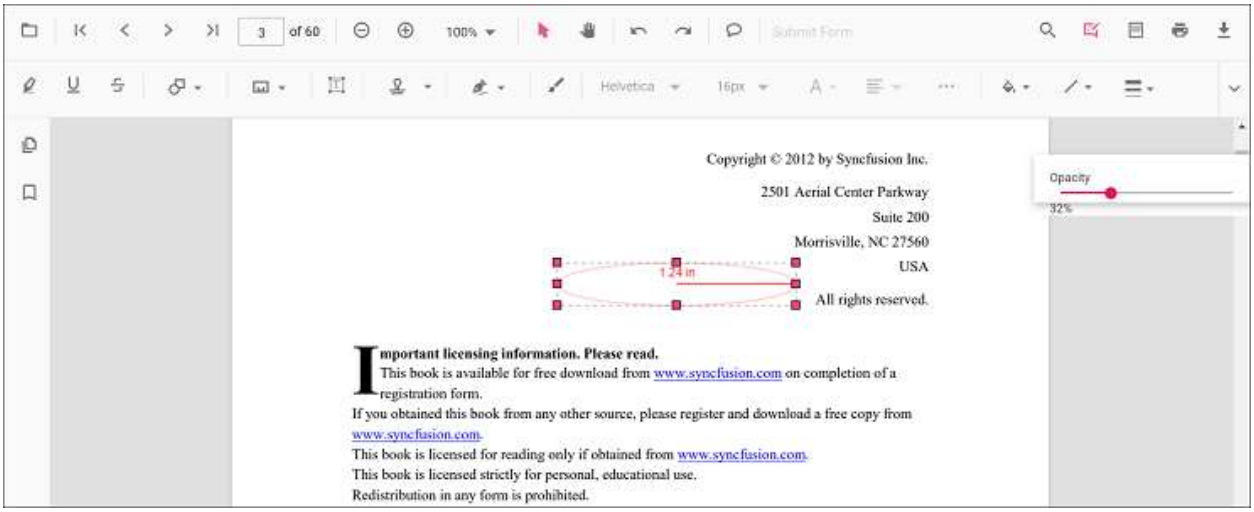
Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit thickness tool.



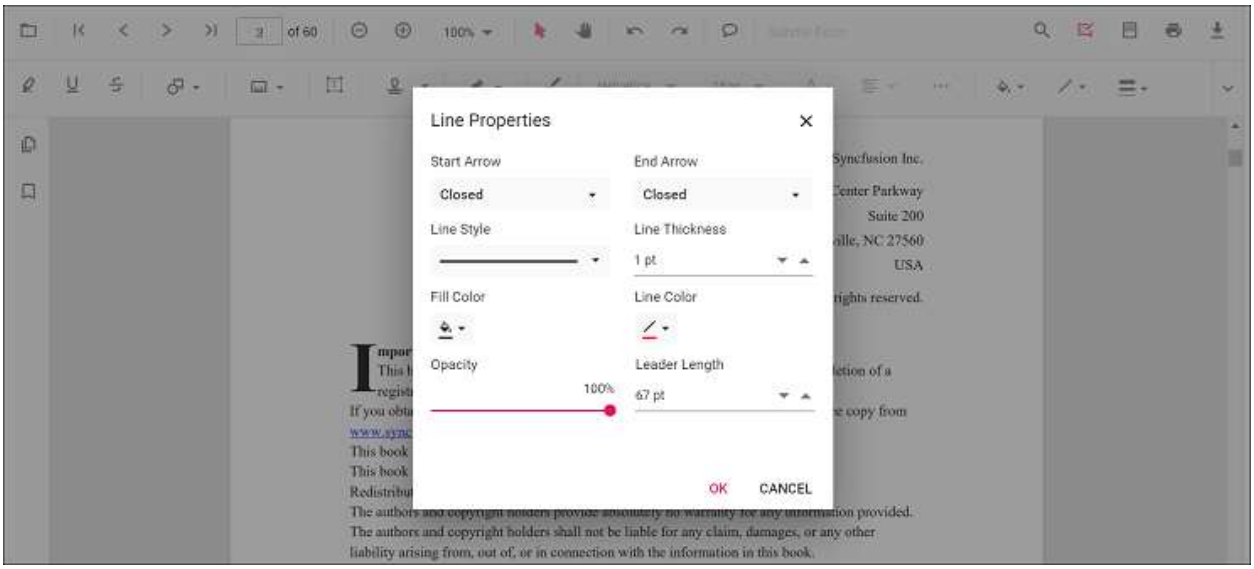
Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Editing the line properties

The properties of the line shapes such as distance and perimeter annotations can be edited using the Line properties window. It can be opened by selecting the Properties option in the context menu that appears on right-clicking the distance and perimeter annotations.



Setting default properties during control initialization

The properties of the shape annotations can be set before creating the control using distanceSettings, perimeterSettings, areaSettings, radiusSettings and volumeSettings. Refer to the following code snippet to set the default annotation settings.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:distanceSettings="distanceSettings"
:perimeterSettings="perimeterSettings"
:areaSettings="areaSettings"
:radiusSettings="radiusSettings"
:volumeSettings="volumeSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
distanceSettings: {fillColor: 'blue', opacity: 0.6, strokeColor: 'green'},
perimeterSettings: {fillColor: 'green', opacity: 0.6, strokeColor: 'blue'},
areaSettings: {fillColor: 'yellow', opacity: 0.6, strokeColor: 'orange'},
radiusSettings: {fillColor: 'orange', opacity: 0.6, strokeColor: 'pink'},
```

```
volumeSettings: {fillColor: 'pink', opacity: 0.6, strokeColor: 'yellow'}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
}
</script>
```

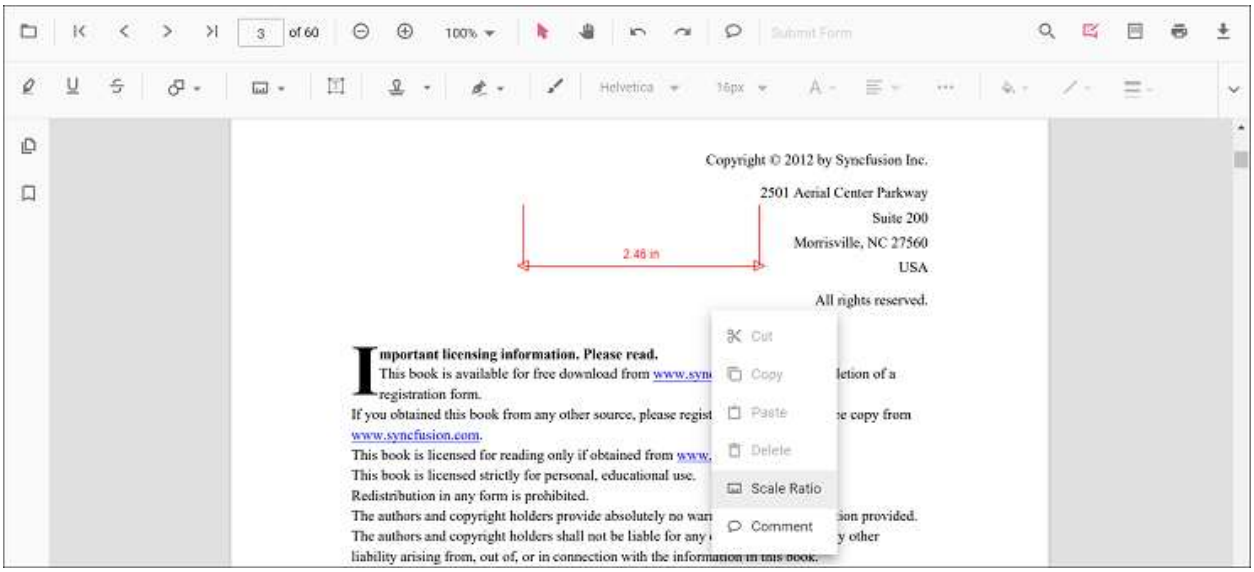
SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:distanceSettings="distanceSettings"
:perimeterSettings="perimeterSettings"
:areaSettings="areaSettings"
:radiusSettings="radiusSettings"
:volumeSettings="volumeSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
distanceSettings: {fillColor: 'blue', opacity: 0.6, strokeColor: 'green'},
perimeterSettings: {fillColor: 'green', opacity: 0.6, strokeColor: 'blue'},
areaSettings: {fillColor: 'yellow', opacity: 0.6, strokeColor: 'orange'},
radiusSettings: {fillColor: 'orange', opacity: 0.6, strokeColor: 'pink'},
volumeSettings: {fillColor: 'pink', opacity: 0.6, strokeColor: 'yellow'}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

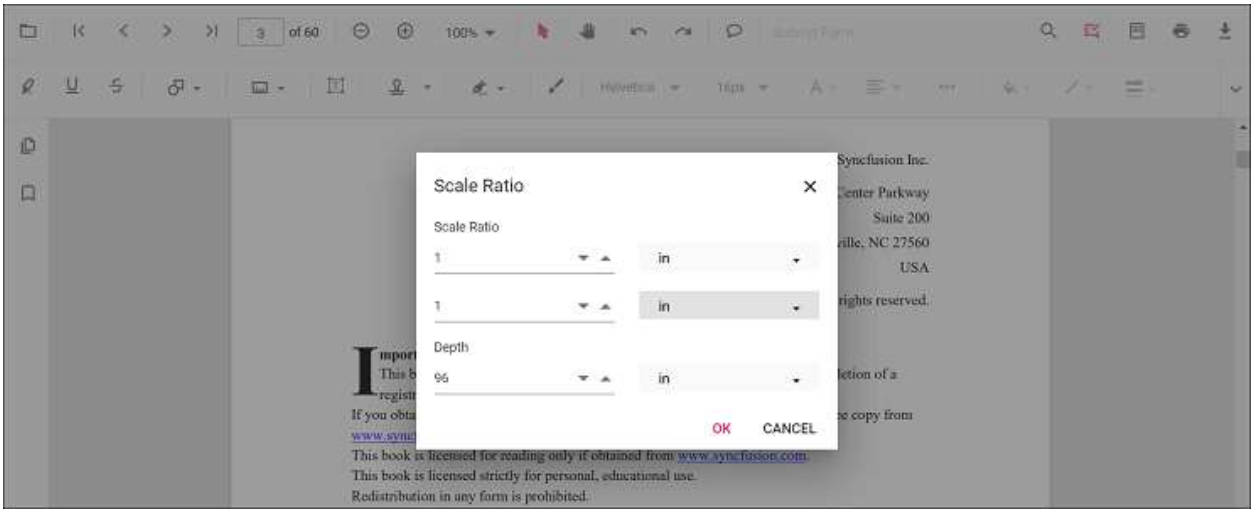
Editing scale ratio and unit of the measurement annotation

The scale ratio and unit of measurement can be modified using the scale ratio option provided in the context menu of the PDF Viewer control.



The Units of measurements support for the measurement annotations in the PDF Viewer are

- Inch
- Millimeter
- Centimeter
- Point
- Pica
- Feet



Setting default scale ratio settings during control initialization

The properties of scale ratio for measurement annotation can be set before creating the control using ScaleRatioSettings as shown in the following code snippet,

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:measurementSettings="measurementSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print ,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
measurementSettings: {scaleRatio: 2, conversionUnit: 'cm', displayUnit:
'cm'}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Distance');
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl"
:measurementSettings="measurementSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print ,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
measurementSettings: {scaleRatio: 2, conversionUnit: 'cm', displayUnit:
'cm'}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Distance');
});
}
}
</script>
```

Free text annotation in Vue Pdfviewer component

The PDF Viewer control provides the options to add, edit, and delete the free text annotations.

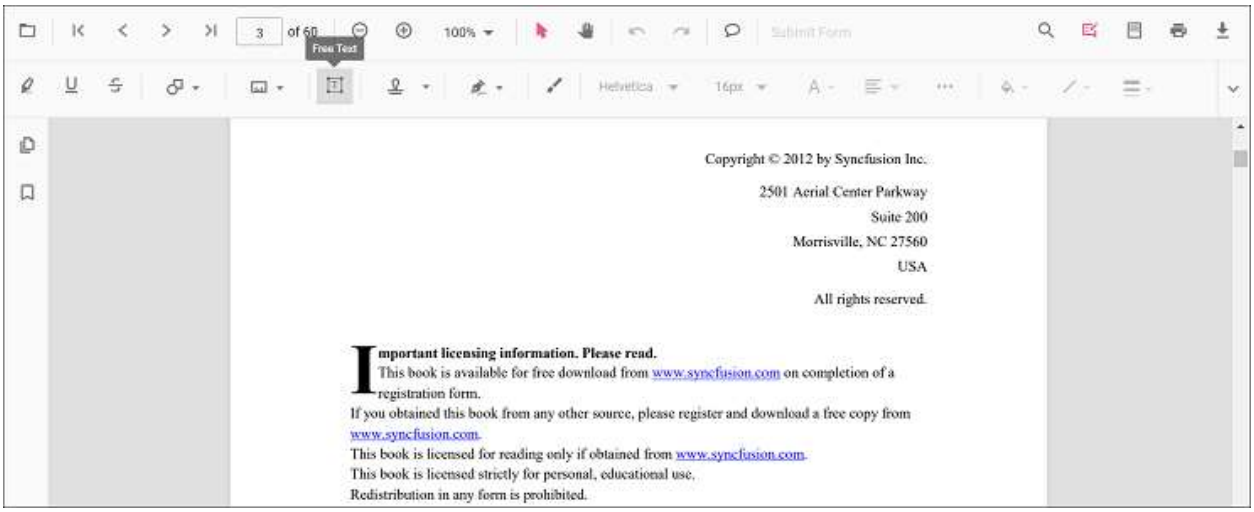
Adding a free text annotation to the PDF document

The Free text annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **Free Text Annotation** button in the annotation toolbar. It enables the Free Text annotation mode.
- You can add the text over the pages of the PDF document.

In the pan mode, if the free text annotation mode is entered, the PDF Viewer control will switch to text select mode.

Copyright © 2001 -2024 Syncfusion Inc.



Refer to the following code sample to switch to the Free Text annotation mode.

STANDALONE

```
<template>
<div id="app">
<button id="set">FreeText</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('FreeText');
```

```
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button id="set">FreeText</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('FreeText');
});
}
}
}
</script>
```

Editing the properties of free text annotation

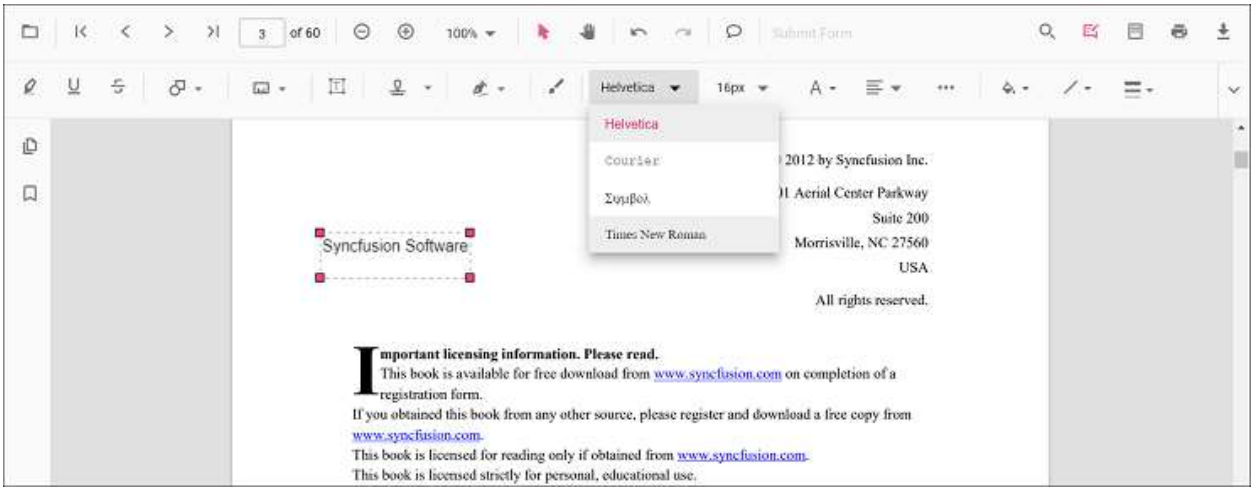
The font family, font size, font styles, font color, text alignment, fill color, the border stroke color, border thickness, and opacity of the free text annotation can be edited using the Font Family tool, Font Size

Copyright © 2001 -2024 Syncfusion Inc.

tool, Font Color tool, Text Align tool, Font Style tool Edit Color tool, Edit Stroke Color tool, Edit Thickness tool, and Edit Opacity tool in the annotation toolbar.

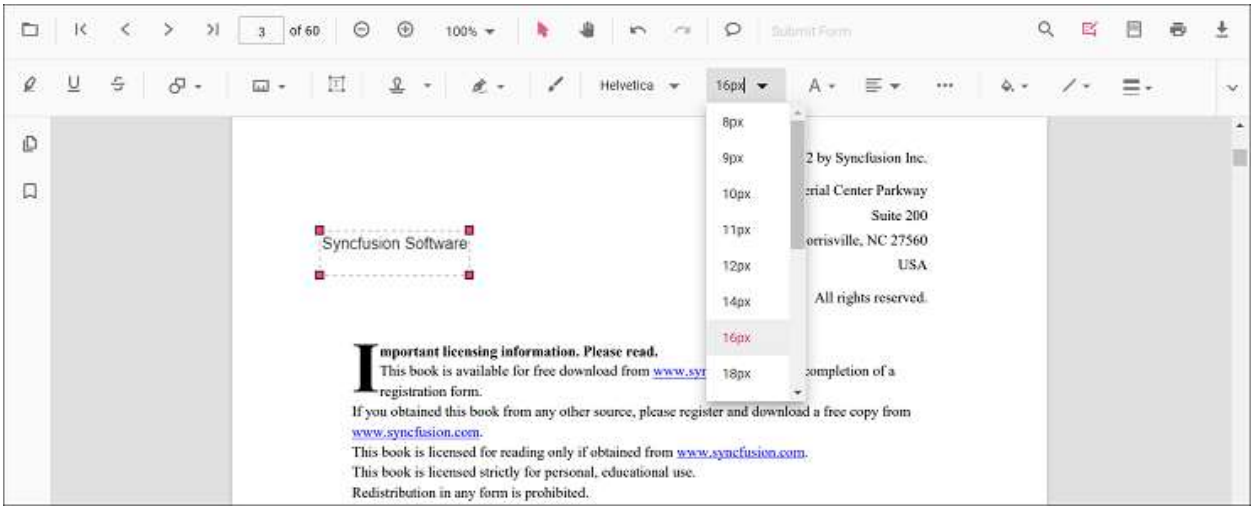
Editing font family

The font family of the annotation can be edited by selecting the desired font in the Font Family tool.



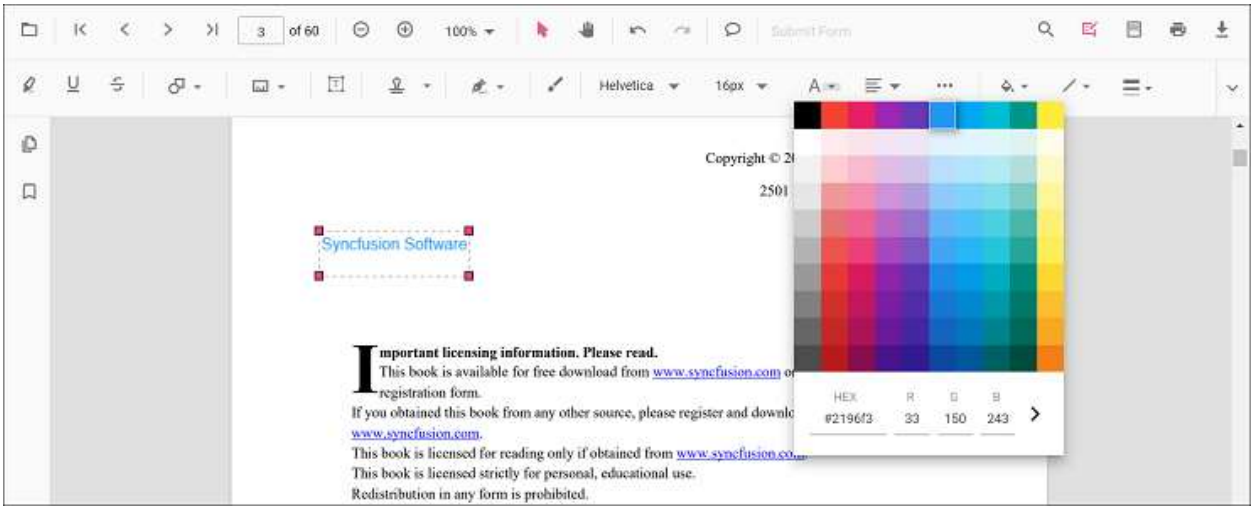
Editing font size

The font size of the annotation can be edited by selecting the desired size in the Font Size tool.



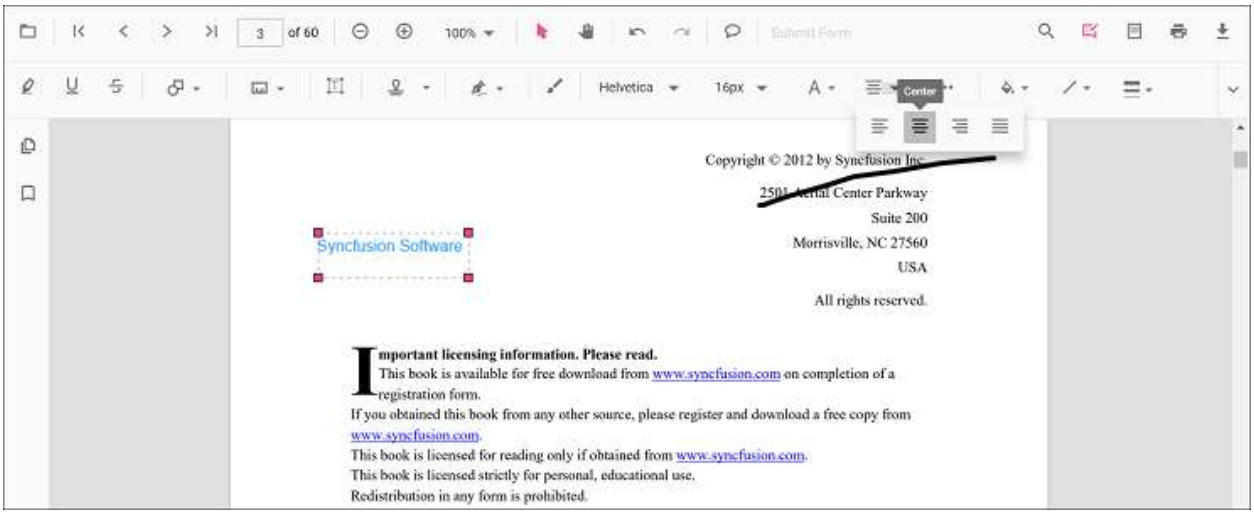
Editing font color

The font color of the annotation can be edited using the color palette provided in the Font Color tool.



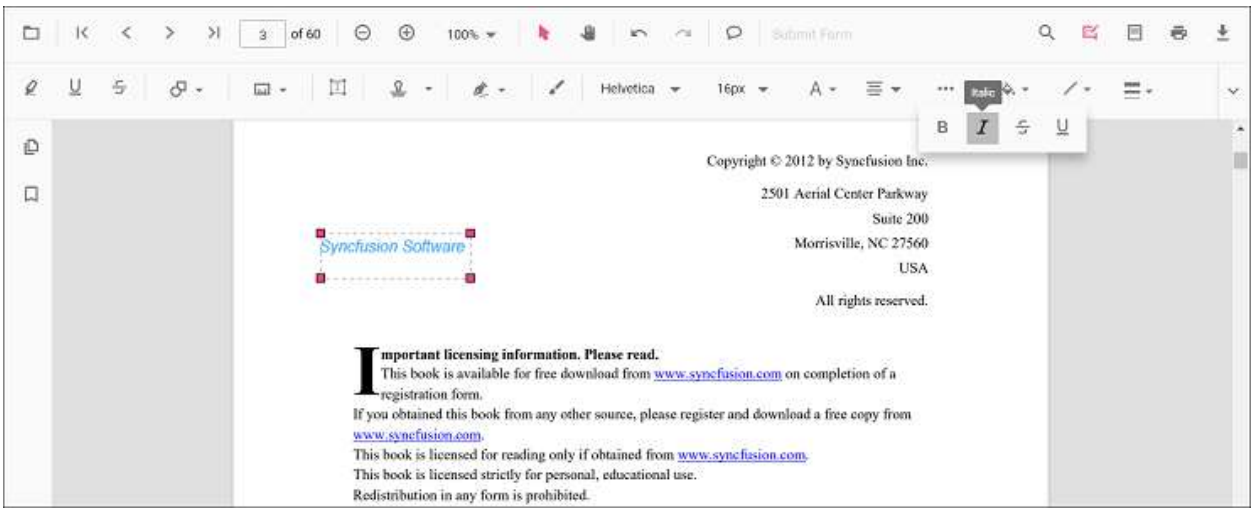
Editing the text alignment

The text in the annotation can be aligned by selecting the desired styles in the drop-down pop-up in the Text Align tool.



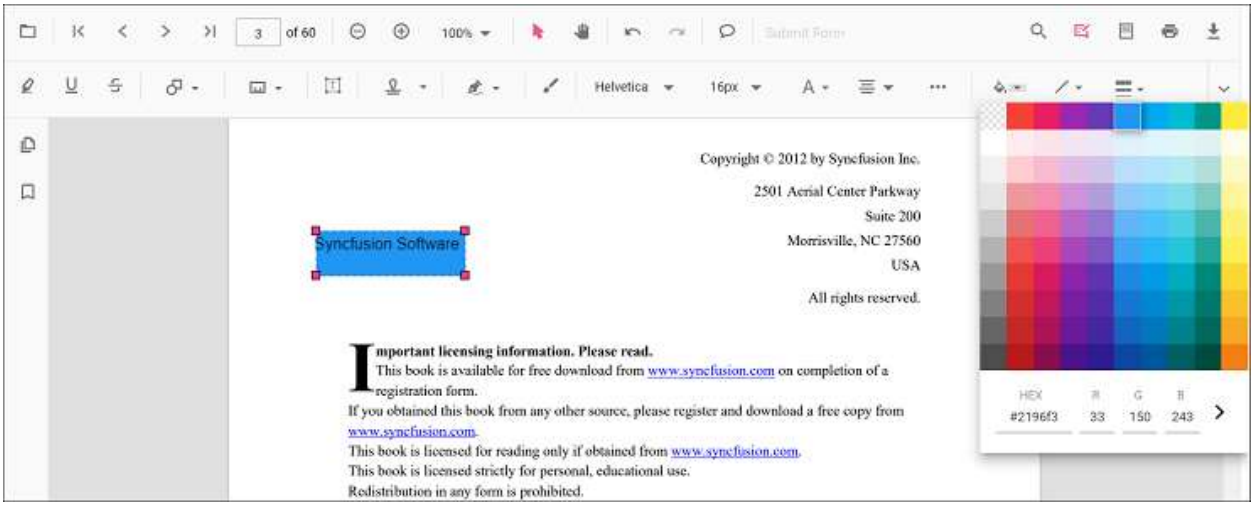
Editing text styles

The style of the text in the annotation can be edited by selecting the desired styles in the drop-down pop-up in the Font Style tool.



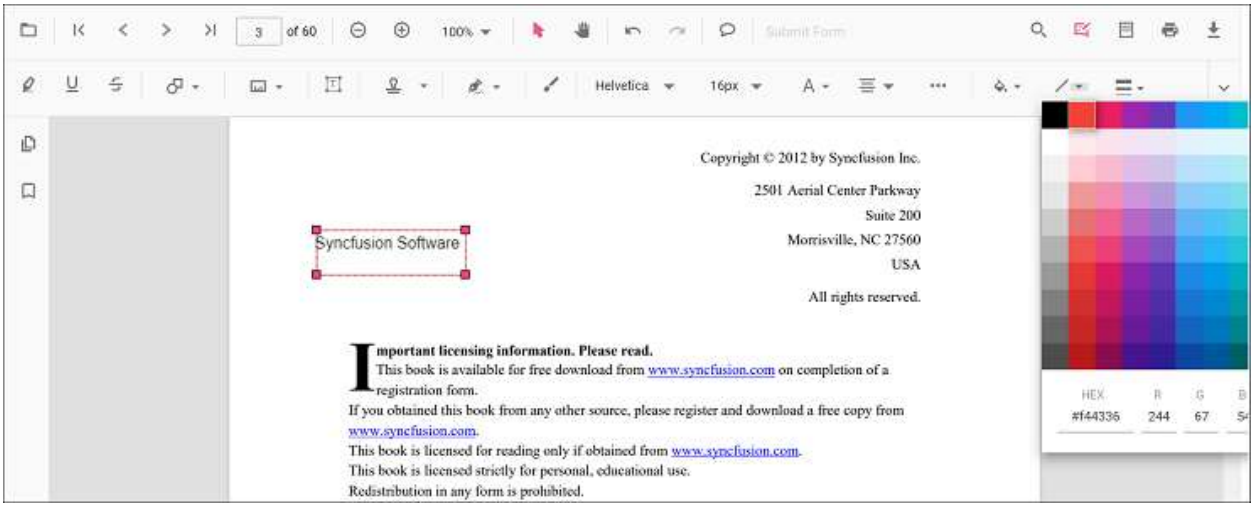
Editing fill color

The fill color of the annotation can be edited using the color palette provided in the Edit Color tool.



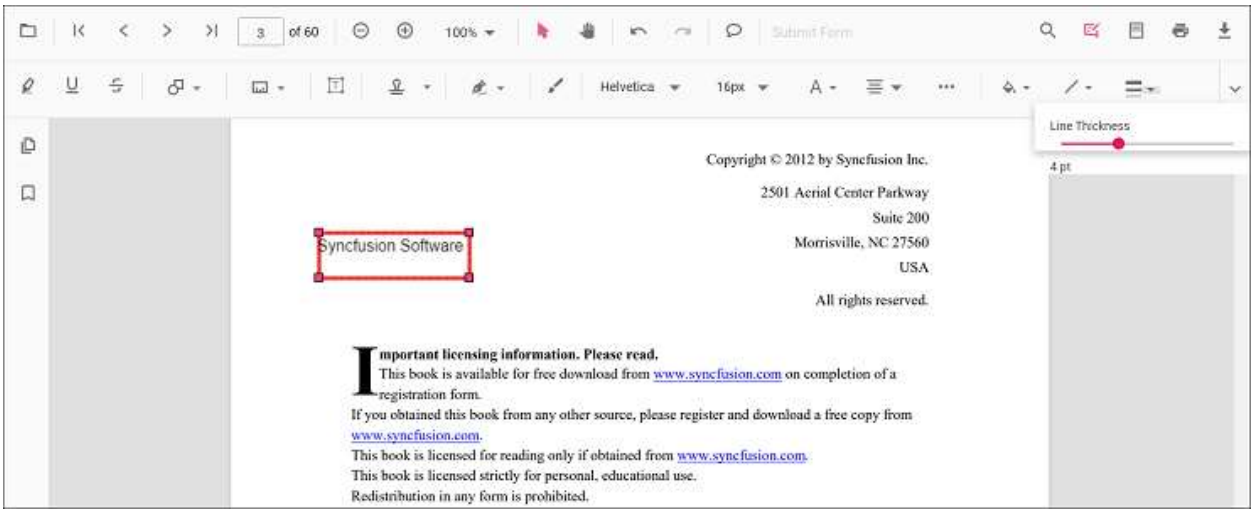
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



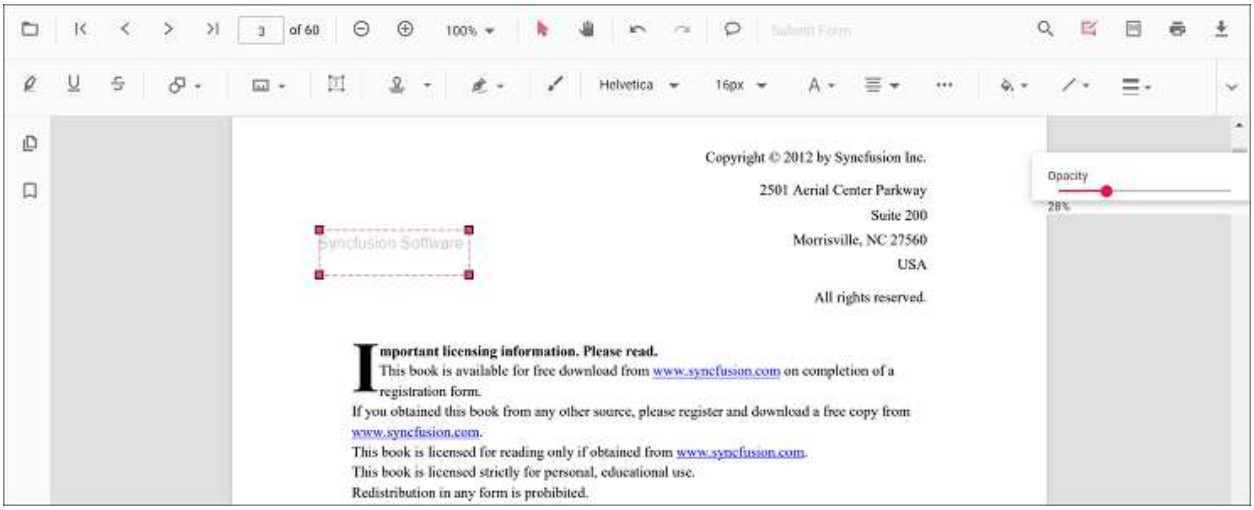
Editing thickness

The border thickness of the annotation can be edited using the range slider provided in the Edit Thickness tool.



Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Setting default properties during control initialization

The properties of the free text annotation can be set before creating the control using the FreeTextSettings.

After editing the default values, they will be changed to the selected values. Refer to the following code sample to set the default free text annotation settings.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:freeTextSettings="freetextSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
freetextSettings: {fillColor: 'green', borderColor: 'blue', fontColor:
'yellow'}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
```

```
}
</script>
```

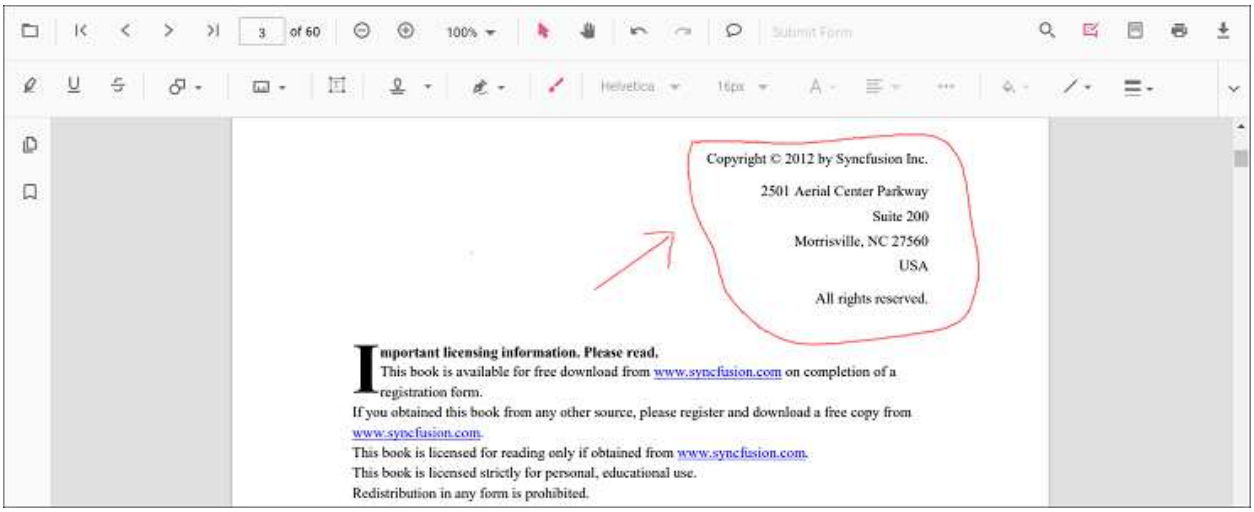
SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl"
:freeTextSettings="freetextSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
freetextSettings: {fillColor: 'green', borderColor: 'blue', fontColor:
'yellow'}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
}
</script>
```

Ink annotation in Vue Pdfviewer component

The PDF Viewer control provides the options to add, edit, and delete the ink annotations.

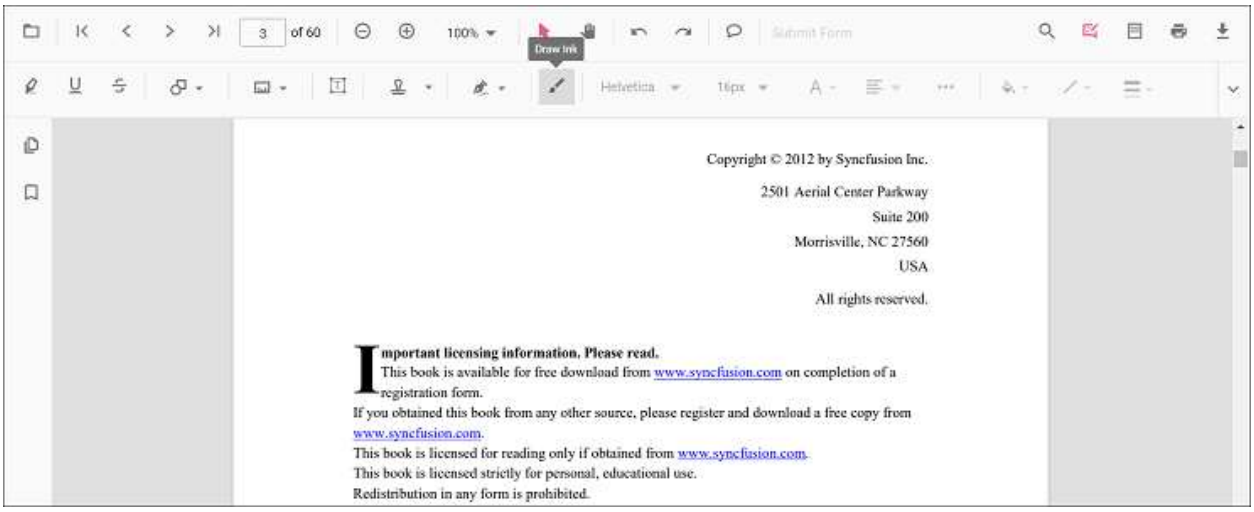
Copyright © 2001 -2024 Syncfusion Inc.



Adding an ink annotation to the PDF document

The ink annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **Draw Ink** button in the annotation toolbar. It enables the ink annotation mode.
- You can draw anything over the pages of the PDF document.



Refer to the following code sample to switch to the ink annotation mode.

STANDALONE

```
<template>
<div id="app">
<button id="set">Draw Ink</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
```

```
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Ink');
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button id="set">Draw Ink</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, ThumbnailView, Print,
TextSelection, TextSearch, Annotation, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
```

Copyright © 2001 -2024 Syncfusion Inc.

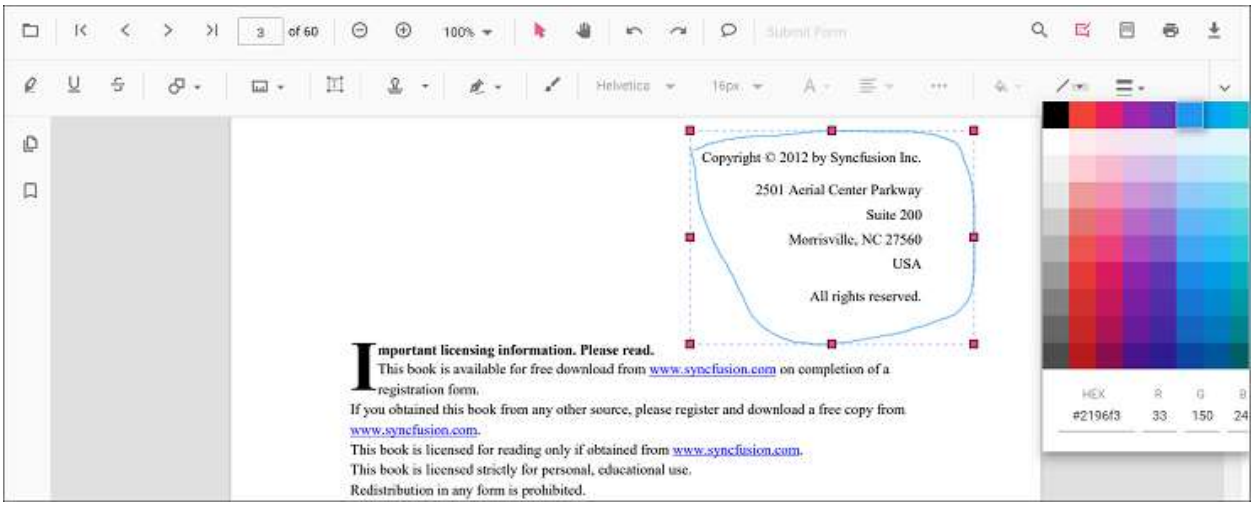
```
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormFields, FormDesigner ]},
methods: {
documentLoad() {
viewer = this.$refs.pdfviewer.ej2Instances;
document.getElementById('set').addEventListener('click', ()=> {
viewer.annotation.setAnnotationMode('Ink');
});
}
}
}
</script>
```

Editing the properties of the ink annotation

The stroke color, thickness, and opacity of the ink annotation can be edited using the Edit stroke color tool, Edit thickness tool, and Edit opacity tool in the annotation toolbar.

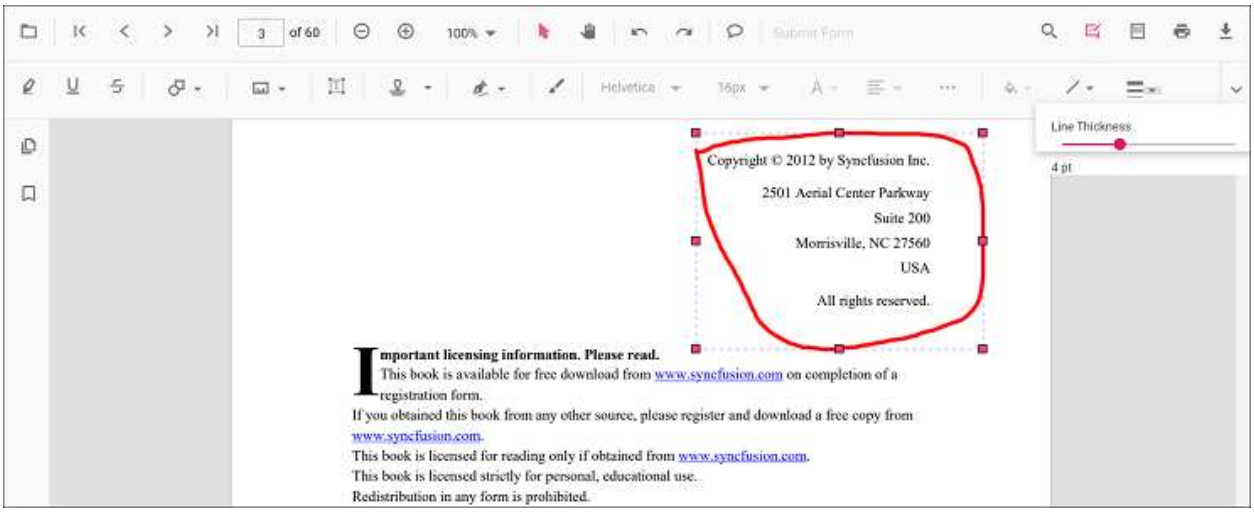
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



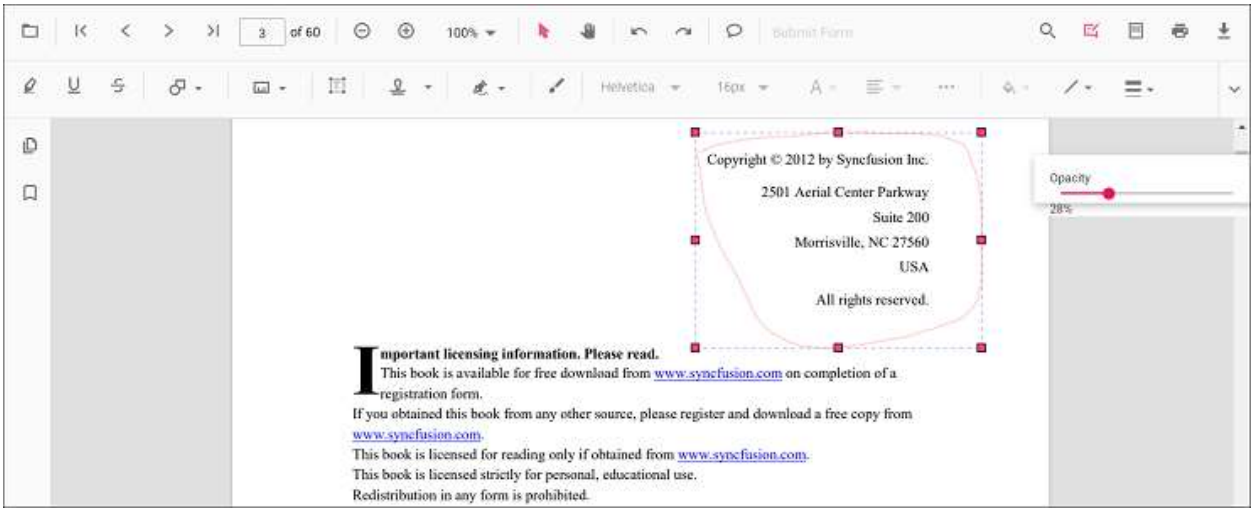
Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit Thickness tool.



Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Setting default properties during the control initialization

The properties of the ink annotation can be set before creating the control using the InkAnnotationSettings.

After editing the default values, they will be changed to the selected values. Refer to the following code sample to set the default ink annotation settings.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:inkAnnotationSettings="inkAnnotationSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
```

```
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
inkAnnotationSettings: {author: 'Syncfusion', strokeColor: 'green',
thickness: 3, opacity: 0.6}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl"
:inkAnnotationSettings="inkAnnotationSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
inkAnnotationSettings: {author: 'Syncfusion', strokeColor: 'green',
thickness: 3, opacity: 0.6}
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
```

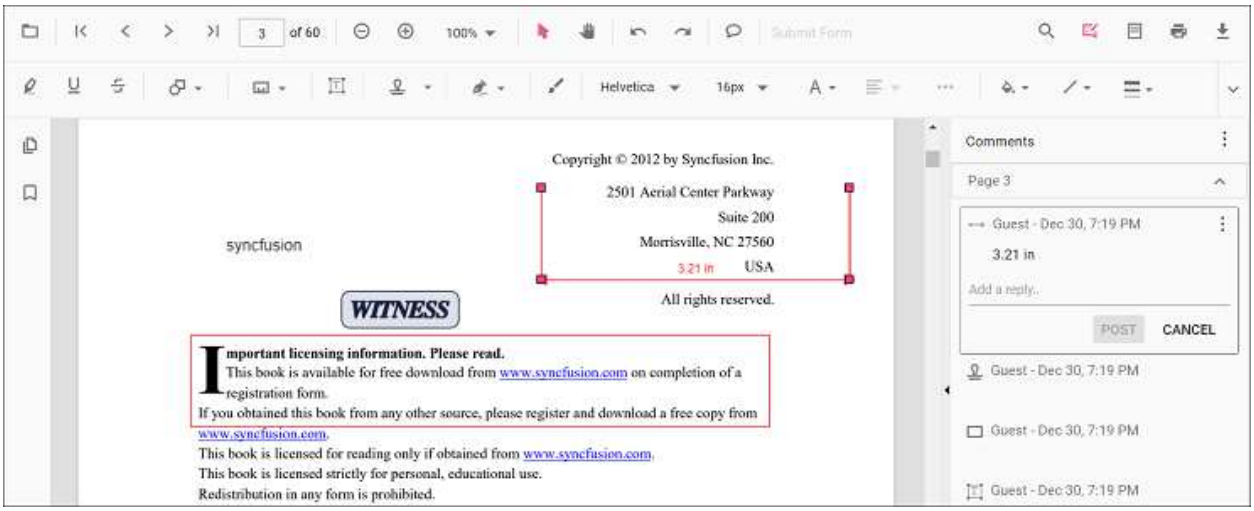
Copyright © 2001 -2024 Syncfusion Inc.

```
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
}
</script>
```

Comments in Vue Pdfviewer component

The PDF Viewer control provides options to add, edit, and delete the comments to the following annotation in the PDF documents:

- Shape annotation
- Stamp annotation
- Sticky note annotation
- Measurement annotation
- Text markup annotation
- Free text annotation
- Ink annotation



Adding a comment to the annotation

Annotation comment, comment replies, and status can be added to the PDF document using the comment panel.

Comment panel

Annotation comments can be added to the PDF using the comment panel. The comment panel can be opened in the following ways:

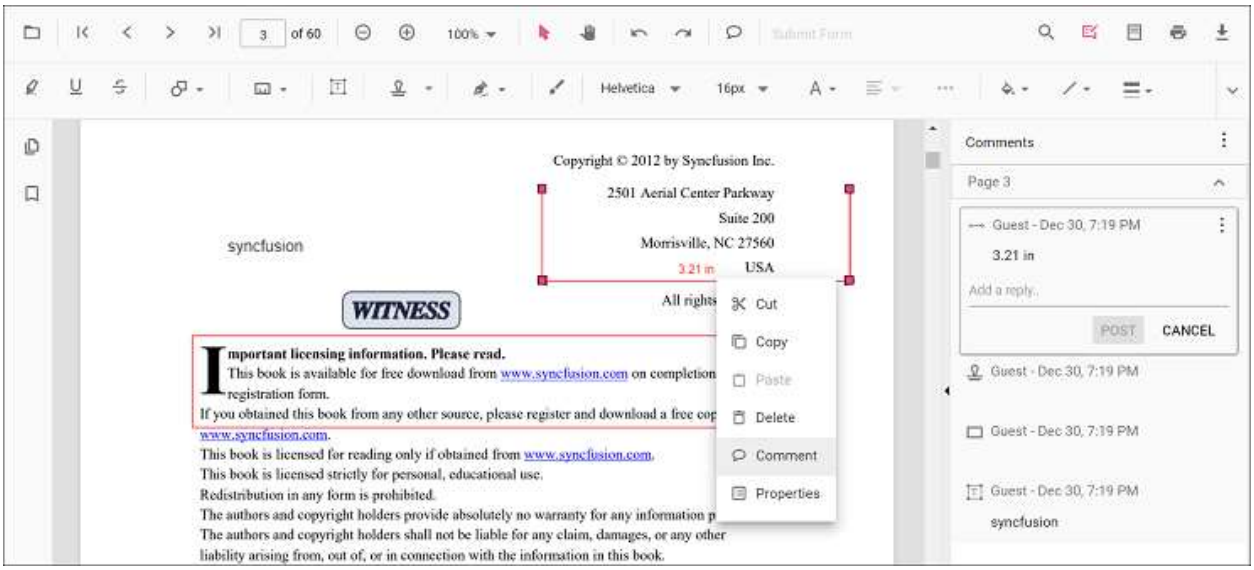
1. Using the annotation menu
    - Click the Edit Annotation button in the PDF Viewer toolbar. A toolbar appears below it.
    - Click the Comment Panel button. A comment panel will appear.
2. Using Context menu
    - Select annotation in the PDF document and right-click it.
    - Select the comment option in the context menu that appears.
3. Using the Mouse click
    - Select annotation in the PDF document and double click it, a comment panel will appear.

Copyright © 2001 -2024 Syncfusion Inc.

If the comment panel is already in the open state, you can select the annotations and add annotation comments using the comment panel.

Adding comments

- Select annotation in the PDF document and click it.
- The selected annotation comment container is highlighted in the comment panel.
- Now, you can add comment and comment replies using the comment panel.

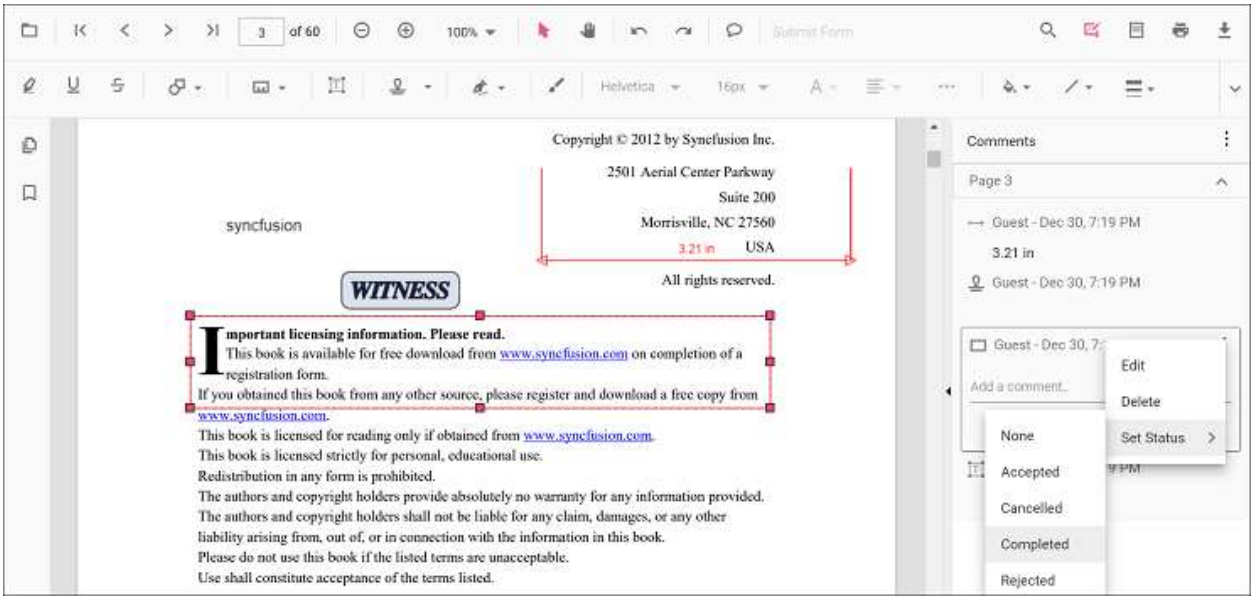


Adding Comment Replies

- The PDF Viewer control provides an option to add multiple replies to the comment.
- After adding the annotation comment, you can add a reply to the comment.

Adding Comment or Reply Status

- Select the Annotation Comments in the comment panel.
- Click the more options button showing in the Comments or reply container.
- Select the Set Status option in the context menu that appears.
- Select the status of the annotation comment in the context menu that appears.



Editing the comments and comments replies of the annotations

The comment, comment replies, and status of the annotation can be edited using the comment panel.

Editing the Comment or Comment Replies

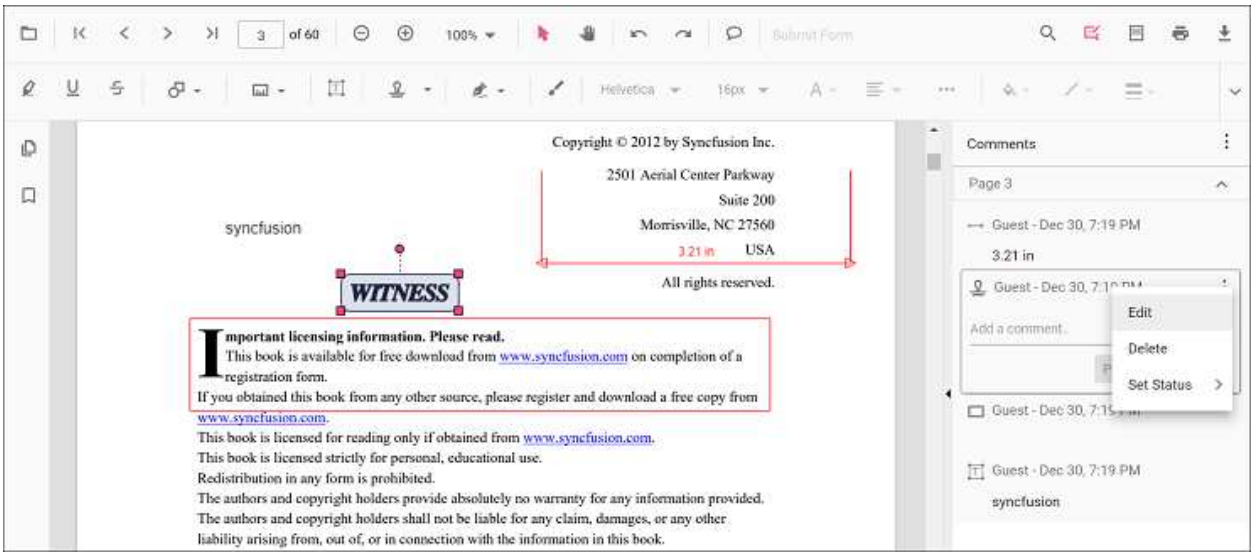
The annotation comment and comment replies can be edited in the following ways:

1. Using the Context menu
    - Select the Annotation Comments in the comment panel.
    - Click the More options button showing in the Comments or reply container.
    - Select the Edit option in the context menu that appears.
    - Now, an editable text box appears. You can change the content of the annotation comment or comment reply.
2. Using the Mouse Click
    - Select the annotation comments in the comment panel.
    - Double click the comment or comment reply content.
    - Now, an editable text box appears. You can change the content of the annotation comment or comment reply.

Editing Comment or Reply Status

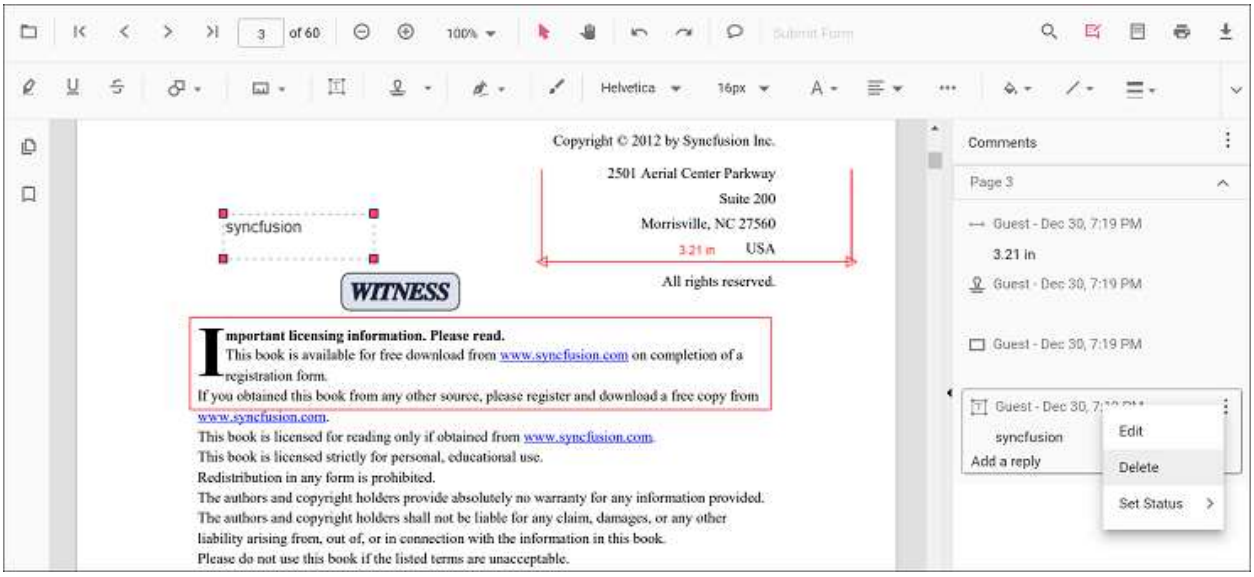
- Select the Annotation Comments in the comment panel.
- Click the more options button showing in the Comments or reply container.
- Select the Set Status option in the context menu that appears.
- Select the status of the annotation comment in the context menu that appears.
- Status ‘None’ is the default state. If the status is set to ‘None,’ the comments or reply does not appear.

Copyright © 2001 -2024 Syncfusion Inc.



Delete Comment or Comment Replies

- Select the Annotation Comments in the comment panel.
- Click the more options button shown in the Comments or reply container.
- Select the Delete option in the context menu that appears.



The annotation will be deleted on deleting the comment using comment panel.

Interaction mode in Vue Pdfviewer component

The PDF Viewer provides interaction mode for easy interaction with the loaded PDF document. Selection mode and panning mode are the two interactions modes.

Selection mode

In this mode, the text selection can be performed in the PDF document loaded in PDF Viewer. The panning and scrolling of the pages by touch cannot be performed in this mode. It allows users to select

and copy text from the PDF files. This is helpful for copying and sharing text content. You can enable/disable the text selection using the following code snippet.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:enableTextSelection="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

SERVER-BACKED

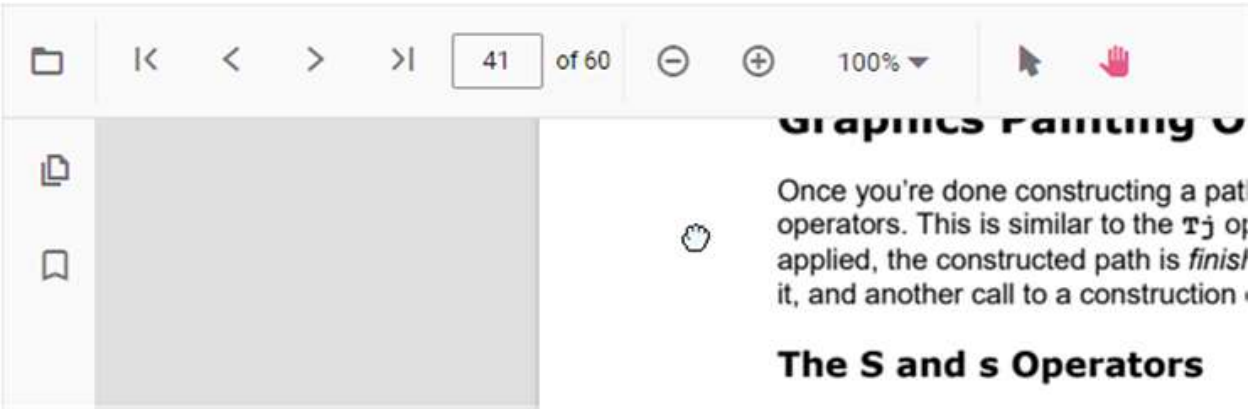
```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:enableTextSelection="true">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

Panning Mode

In this mode, the panning and scrolling of the pages by touch can be performed in the PDF document loaded in the PDF Viewer, but the text selection cannot be performed.



You can switch the interaction mode of PDF Viewer by using the following code snippet.,

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:enableTextSelection="false"
:interactionMode="interactionMode">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
```

```
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
interactionMode:"InteractionMode.Pan"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:enableTextSelection="false"
:interactionMode="interactionMode">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
interactionMode:"InteractionMode.Pan"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,
Annotation, ThumbnailView, Print, TextSelection, TextSearch ]},
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

See also

- Toolbar items
- Feature Modules

Form Designer

Create form fields programmatically in Vue Pdfviewer component

The PDF Viewer control provides the option to add, edit and delete the Form Fields. The Form Fields type supported by the PDF Viewer Control are:

- Textbox
- Password
- CheckBox
- RadioButton
- ListBox
- DropDown
- SignatureField
- InitialField

Add a form field to PDF document programmatically

Using addFormField method, the form fields can be added to the PDF document programmatically. We need to pass two parameters in this method. They are Form Field Type and Properties of Form Field Type. To add form field programmatically, Use the following code.

APP.VUE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad"
:resourceUrl="resourceUrl">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
resourceUrl:"https://cdn.syncfusion.com/ej2/23.1.43/dist/ej2-pdfviewer-lib"
};
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.formDesignerModule.addFormField("Textbox", { name: "First Name",
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }});
viewer.formDesignerModule.addFormField("Textbox", { name: "Middle Name",
bounds: { X: 338, Y: 229, Width: 150, Height: 24 }});
viewer.formDesignerModule.addFormField('Textbox', {name: 'Last Name',bounds:
{ X: 530, Y: 229, Width: 150, Height: 24 },});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 148, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 292, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Month',bounds:
{ X: 146, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Date',bounds:
{ X: 193, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Year',bounds:
{ X: 242, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('InitialField', {name:
'Agree',bounds: { X: 148, Y: 408, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('InitialField', {name: 'Do Not
Agree',bounds: { X: 148, Y: 466, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Text
Message',bounds: { X: 56, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Email',bounds: {
X: 242, Y: 664, Width: 20, Height: 20 },isChecked: false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Appointment
Reminder',bounds: { X: 56, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Request for
Customerservice',bounds: { X: 56, Y: 778, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Information
Billing',bounds: { X: 290, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Email',bounds:
{ X: 146, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Phone',bounds:
{ X: 482, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('SignatureField', {name:
'Sign',bounds: { X: 57, Y: 923, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Month',bounds:
{ X: 386, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Date',bounds:
{ X: 434, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Year',bounds:
{ X: 482, Y: 923, Width: 35, Height: 24 },});
}
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
```

Note: To set up the **server-backed PDF Viewer** in the app.vue file, include the following serviceUrl:

serviceUrl: “https://services.syncfusion.com/vue/production/api/pdfviewer”

Within the template, configure the PDF Viewer by adding the :serviceUrl=“serviceUrl” attribute inside the element.

{% previewsample "page.domainurl/code-snippet/pdfviewer/addformfield-cs1" %}

Edit or update form field programmatically

Using updateFormField method, Form Field can be updated programmatically. We should get the Form Field object/Id from FormFieldCollections property that you would like to edit and pass it as a parameter to updateFormField method. The second parameter should be the properties that you would like to update for Form Field programmatically. We have updated the value and background Color properties of Textbox Form Field.

APP.VUE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad"
:resourceUrl="resourceUrl">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
resourceUrl:"https://cdn.syncfusion.com/ej2/23.1.43/dist/ej2-pdfviewer-lib"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.formDesignerModule.addFormField("Textbox", { name: "First Name",
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }});
```

```
viewer.formDesignerModule.addFormField("Textbox", { name: "Middle Name",
bounds: { X: 338, Y: 229, Width: 150, Height: 24 }});
viewer.formDesignerModule.addFormField('Textbox', {name: 'Last Name',bounds:
{ X: 530, Y: 229, Width: 150, Height: 24 },});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 148, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 292, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Month',bounds:
{ X: 146, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Date',bounds:
{ X: 193, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Year',bounds:
{ X: 242, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('InitialField', {name:
'Agree',bounds: { X: 148, Y: 408, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('InitialField', {name: 'Do Not
Agree',bounds: { X: 148, Y: 466, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Text
Message',bounds: { X: 56, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Email',bounds: {
X: 242, Y: 664, Width: 20, Height: 20 },isChecked: false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Appointment
Reminder',bounds: { X: 56, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Request for
Customerservice',bounds: { X: 56, Y: 778, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Information
Billing',bounds: { X: 290, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Email',bounds:
{ X: 146, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Phone',bounds:
{ X: 482, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('SignatureField', {name:
'Sign',bounds: { X: 57, Y: 923, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Month',bounds:
{ X: 386, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Date',bounds:
{ X: 434, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Year',bounds:
{ X: 482, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.updateFormField(viewer.formFieldCollections[0], {
backgroundColor: 'red' });
}
}
}
</script>
```

Note: To set up the **server-backed PDF Viewer** in the app.vue file, include the following serviceUrl:

serviceUrl: “https://services.syncfusion.com/vue/production/api/pdfviewer”

Copyright © 2001 -2024 Syncfusion Inc.

Within the template, configure the PDF Viewer by adding the :serviceUrl="serviceUrl" attribute inside the element.

{% previewsample "page.domainurl/code-snippet/pdfviewer/updateformfield-cs1" %}

Delete form field programmatically

Using deleteFormField method, the form field can be deleted programmatically. We should retrieve the Form Field object/Id from FormFieldCollections property that you would like to delete and pass it as a parameter to deleteFormField method. To delete a Form Field programmatically, use the following code.

APP.VUE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad"
:resourceUrl="resourceUrl">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
resourceUrl:"https://cdn.syncfusion.com/ej2/23.1.43/dist/ej2-pdfviewer-lib"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.formDesignerModule.addFormField("Textbox", { name: "First Name",
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }});
viewer.formDesignerModule.addFormField("Textbox", { name: "Middle Name",
bounds: { X: 338, Y: 229, Width: 150, Height: 24 }});
viewer.formDesignerModule.addFormField('Textbox', {name: 'Last Name',bounds:
{ X: 530, Y: 229, Width: 150, Height: 24 },});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 148, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
viewer.formDesignerModule.addFormField('RadioButton', {bounds: { X: 292, Y:
289, Width: 18, Height: 18 },name: 'Gender',isSelected: false,});
```

Copyright © 2001 -2024 Syncfusion Inc.

```
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Month',bounds:
{ X: 146, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Date',bounds:
{ X: 193, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOB Year',bounds:
{ X: 242, Y: 320, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('InitialField', {name:
'Agree',bounds: { X: 148, Y: 408, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('InitialField', {name: 'Do Not
Agree',bounds: { X: 148, Y: 466, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Text
Message',bounds: { X: 56, Y: 664, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Email',bounds: {
X: 242, Y: 664, Width: 20, Height: 20 },isChecked: false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Appointment
Reminder',bounds: { X: 56, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Request for
Customerservice',bounds: { X: 56, Y: 778, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('CheckBox', {name: 'Information
Billing',bounds: { X: 290, Y: 740, Width: 20, Height: 20 },isChecked:
false,});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Email',bounds:
{ X: 146, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'My Phone',bounds:
{ X: 482, Y: 850, Width: 200, Height: 24 },});
viewer.formDesignerModule.addFormField('SignatureField', {name:
'Sign',bounds: { X: 57, Y: 923, Width: 200, Height: 43 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Month',bounds:
{ X: 386, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Date',bounds:
{ X: 434, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.addFormField('Textbox', {name: 'DOS Year',bounds:
{ X: 482, Y: 923, Width: 35, Height: 24 },});
viewer.formDesignerModule.deleteFormField(viewer.formFieldCollections[0]);
}
}
}
</script>
```

Note: To set up the **server-backed PDF Viewer** in the app.vue file, include the following serviceUrl:

serviceUrl: “https://services.syncfusion.com/vue/production/api/pdfviewer”

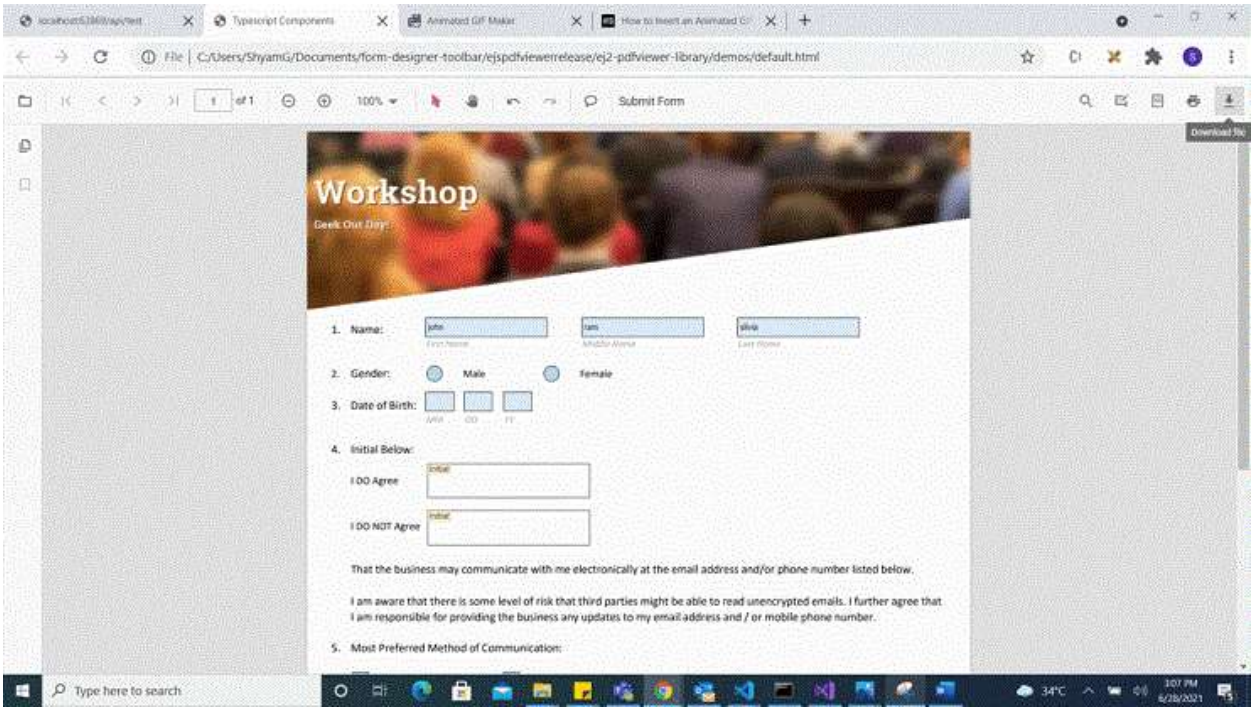
Within the template, configure the PDF Viewer by adding the :serviceUrl=“serviceUrl” attribute inside the element.

{% previewsample "page.domainurl/code-snippet/pdfviewer/deleteformfield-cs1" %}

Saving the form fields

When the download icon is selected on the toolbar, the Form Fields will be saved in the PDF document and this action will not affect the original document. Refer the below GIF for further reference.

Copyright © 2001 -2024 Syncfusion Inc.



You can invoke download action using following code snippet.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="downloadClicked">Download</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
```

```
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
methods: {
downloadClicked: function (args) {
viewer.download();
},
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
},
}
}
</script>
```

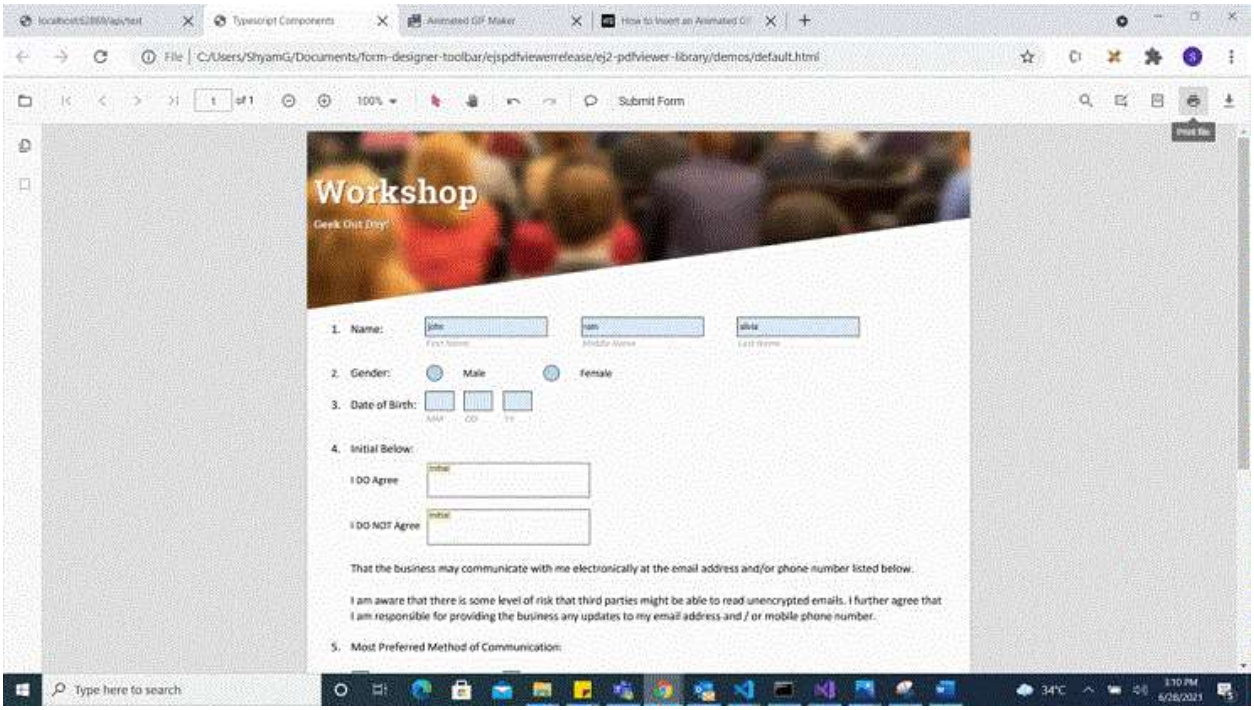
SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="downloadClicked">Download</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
methods: {
downloadClicked: function (args) {
viewer.download();
},
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
},
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

Printing the form fields

When the print icon is selected on the toolbar, the PDF document will be printed along with the Form Fields added to the pages and this action will not affect the original document. Refer the below GIF for further reference.



You can invoke print action using the following code snippet.,

STANDALONE

```
<template>
<div id="app">
<button v-on:click="printClicked">Print</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
methods: {
printClicked: function (args) {
viewer.print.print();
},
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
},
}
}
</script>
```

SERVER-BACKED

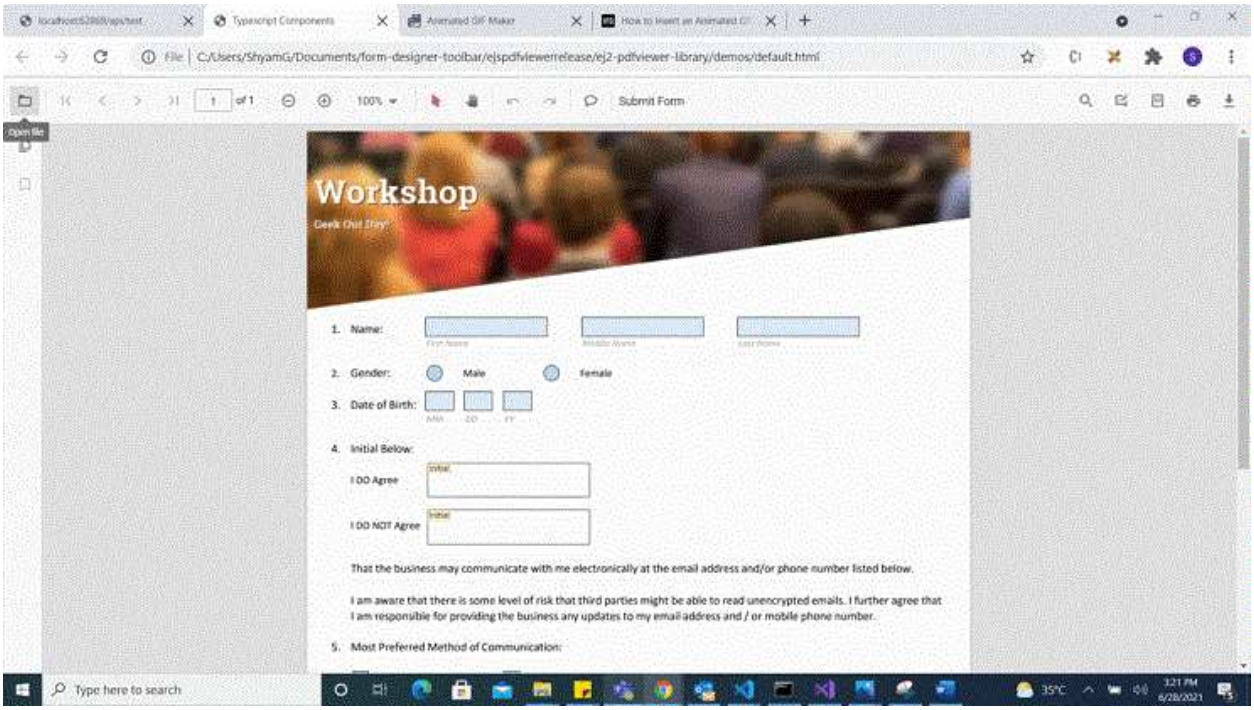
```
<template>
<div id="app">
<button v-on:click="printClicked">Print</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
methods: {
printClicked: function (args) {
viewer.print.print();
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
},
}
}
</script>
```

Open the existing PDF document

We can open the already saved PDF document contains Form Fields in it by clicking the open icon in the toolbar. Refer the below GIF for further reference.



Validate form fields

The form fields in the PDF Document will be validated when the enableFormFieldsValidation is set to true and hook the validateFormFields. The validateFormFields will be triggered when the PDF document is downloaded or printed with the non-filled form fields. The non-filled fields will be obtained in the nonFillableFields property of the event arguments of validateFormFields.

Add the following code snippet to validate the form fields,

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad"
:enableFormFieldsValidation="true"
:validateFormFields="validateFormFields">
</ejs-pdfviewer>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.formDesignerModule.addFormField("Textbox", { name: "Textbox", bounds:
{ X: 70, Y: 229, Width: 150, Height: 24 } });
},
validateFormFields: function(args){
var nonfilledFormFields = args.nonFillableFields
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-vue-pdfviewer/styles/material.css';
#pdfViewer {
height: 640px;
}
</style>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
```

Copyright © 2001 -2024 Syncfusion Inc.

```
:documentPath="documentPath"
:documentLoad="documentLoad"
:enableFormFieldsValidation="true"
:validateFormFields="validateFormFields">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.formDesignerModule.addFormField("Textbox", { name: "Textbox", bounds:
{ X: 70, Y: 229, Width: 150, Height: 24 } });
},
validateFormFields: function(args){
var nonfilledFormFields = args.nonFillableFields
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-vue-pdfviewer/styles/material.css';
#pdfViewer {
height: 640px;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Export and import form fields

The PDF Viewer control provides the support to export and import the form field data in the following formats using the methods importFormFields, exportFormFields, exportFormFieldsAsObject.

- FDF
- XFDF
- JSON
- XML

Export and import as FDF

Using the exportFormFields method, the form field data can be exported in the specified data format. This method accepts two parameters:

- The first one must be the destination path for the exported data. If path is not specified, it will ask for the location while exporting.
- The second parameter should be the format type of the form data.

The following code explains how to export and import the form field data as FDF.

```ts
<template>
<div id="app">
<button v-on:click="exportFdf">Export FDF</button>
<button v-on:click="imporFdf">Import FDF</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,ThumbnailView, Print,TextSelection, TextSearch,
Annotation, FormDesigner, FormFields, FormFieldDataFormat } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
```

```
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner ]},
methods: {
// Event triggers on the Export FDF button click.
exportFdf: function () {
// Data must be the desired path for the exported document.
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.exportFormFields('Data', FormFieldDataFormat.Fdf);
},
// Event triggers on the Import FDF button click.
importFdf: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
// The file for importing the form fields should be placed in the desired location and the path should be provided correctly
viewer.importFormFields('File',FormFieldDataFormat.Fdf);
},
}
}
</script>
`
```

Export and import as XFDF

The following code explains how to export and import the form field data as XFDF.

```
`ts
<template>
<div id="app">
<button v-on:click="exportXfdf">Export XFDF</button>
```

```
<button v-on:click="importXfdf">Import XFDF</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,ThumbnailView, Print,TextSelection, TextSearch,
Annotation, FormDesigner, FormFields, FormFieldDataFormat } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner ]},
methods: {
// Event triggers on the Export XFDF button click.
exportXfdf: function () {
// Data must be the desired path for the exported document.
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.exportFormFields('Data', FormFieldDataFormat.Xfdf);
},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
// Event triggers on the Import XFDF button click.
importXfdf: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
// The file for importing the form fields should be placed in the desired location and the path should be provided correctly
viewer.importFormFields('File',FormFieldDataFormat.Xfdf);
},
}
}
</script>
`
```

Export and import as JSON

The following code explains how to export and import the form field data as JSON.

```
`ts
<template>
<div id="app">
<button v-on:click="exportJson">Export JSON</button>
<button v-on:click="importJson">Import JSON</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,ThumbnailView, Print,TextSelection, TextSearch,
Annotation, FormDesigner, FormFields, FormFieldDataFormat } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
```

```
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner ]},
methods: {
// Event triggers on the Export JSON button click.
exportJson: function () {
// Data must be the desired path for the exported document.
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.exportFormFields('Data', FormFieldDataFormat.Json);
},
// Event triggers on the Import JSON button click.
importJson: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
// The file for importing the form fields should be placed in the desired location and the path should be provided correctly
viewer.importFormFields('File',FormFieldDataFormat.Json);
},
}
}
`
```

Export and import as Object

The PDF Viewer control supports exporting the form field data as an object, and the exported data will be imported into the current PDF document from the object.

The following code shows how to export the form field data as an object and import the form field data from that object into the current PDF document via a button click.

```ts
<template>
<div id="app">
```

```
<button v-on:click="exportDataAsObject">Export Object</button>
<button v-on:click="importData">Import Data</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,ThumbnailView, Print,TextSelection, TextSearch,
Annotation, FormDesigner, FormFields, FormFieldDataFormat } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var exportedData;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer"
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner ]},
methods: {
// Event triggers on the Export Object button click.
exportDataAsObject: function (args) {
// Export the form fields data to an FDF object.
exportedData = viewer.exportFormFieldsAsObject(FormFieldDataFormat.Fdf);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
//// Export the form fields data to an XFDF object.
//exportedData = viewer.exportFormFieldsAsObject(FormFieldDataFormat.Xfdf);
//// Export the form fields data to an JSON object.
//exportedData = viewer.exportFormFieldsAsObject(FormFieldDataFormat.Json);
},
// Event triggers on the Import Data button click.
importData: function (args) {
// Import the form fields data from the FDF object into the current PDF document.
viewer.importFormFields(exportedData, FormFieldDataFormat.Fdf);
//// Import the form fields data from the XFDF object into the current PDF document.
//viewer.importFormFields(exportedData, FormFieldDataFormat.Xfdf);
//// Import the form fields data from the JSON object into the current PDF document.
//viewer.importFormFields(exportedData, FormFieldDataFormat.Json);
},
}
}
`
```

Export and import as XML

The following code explains how to export and import the form field data as XML.

```ts
<template>
<div id="app">
<button v-on:click="exportXml">Export XML</button>
<button v-on:click="importXml">Import XML</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
```

```
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,ThumbnailView, Print,TextSelection, TextSearch,
Annotation, FormDesigner, FormFields, FormFieldDataFormat } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation, BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner ]},
methods: {
// Event triggers on the Export XML button click.
exportXml: function () {
// Data must be the desired path for the exported document.
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.exportFormFields('Data', FormFieldDataFormat.Xml);
},
// Event triggers on the Import XML button click.
importXml: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
// The file for importing the form fields should be placed in the desired location and the path should be
provided correctly
viewer.importFormFields('File',FormFieldDataFormat.Xml);
},
}
}
`
```

Copyright © 2001 -2024 Syncfusion Inc.

Form field properties

Form field properties in Syncfusion PDF Viewer allow you to customize and interact with form fields embedded within PDF documents. This documentation provides an overview of the form field properties supported by the Syncfusion PDF Viewer and explains how to use them effectively.

- Textbox
- Password
- CheckBox
- RadioButton
- ListBox
- DropDown
- SignatureField
- InitialField

Signature and initial fields settings

Using the updateFormField method, the form fields can be updated programmatically.

The following code example explains how to update the signature field properties on a button click.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="updateProperties">Update Properties</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties: function (args) {
```

```
viewer = this.$refs.pdfviewer.ej2Instances;
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Signature',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Signature',
thickness: 4
});
},
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="updateProperties">Update Properties</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
name: 'Signature',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Signature',
thickness: 4
});
},
}
}
</script>
```

The following code example explains how to update the properties of the signature field added to the document from the form designer toolbar.

`ts

// Properties to customize the signature field settings

viewer.signatureFieldSettings = {

// Set the name of the form field element.

name: 'Signatute',

// Specify whether the signature field is in read-only or read-write mode.

isReadOnly: false,

// Set the visibility of the form field.

visibility: 'visible',

// Specify whether the field is mandatory or not.

isRequired: false,

// Specify whether to print the signature field.

isPrint: true,

// Set the text to be displayed as a tooltip.

tooltip: 'Signatute',

// Set the thickness of the signature field. To hide the borders, set the value to 0 (zero).

thickness: 4,

// Specify the properties of the signature Dialog Settings in the signature field.

signatureDialogSettings: {

isplayMode: DisplayMode.Draw | DisplayMode.Upload | DisplayMode.Text,

hideSaveSignature: false,

},

// Specify the properties of the signature indicator in the signature field.

Copyright © 2001 -2024 Syncfusion Inc.

signatureIndicatorSettings: {

opacity: 1,

backgroundColor: '#237ba2',

height: 50,

fontSize: 15,

text: 'Signature Field',

color: 'white'

},

};

`

The following code example explains how to update the properties of the initial field added to the document from the form designer toolbar.

`ts

// Properties to customize the initial field settings

viewer.initialFieldSettings = {

// Set the name of the form field element.

name: 'Initial',

// Specify whether the initial field is in read-only or read-write mode.

isReadOnly: false,

// Set the visibility of the form field.

visibility: 'visible',

// Specify whether the field is mandatory or not.

isRequired: false,

// Specify whether to print the initial field.

isPrint: true,

// Set the text to be displayed as tooltip.

tooltip: 'Initial',

// Set the thickness of the initial field. To hide the borders, set the value to 0 (zero).

thickness: 4,

// Specify the properties of the initial indicator in the initial field.

Copyright © 2001 -2024 Syncfusion Inc.

initialIndicatorSettings: {

opacity: 1,

backgroundColor: '#237ba2',

height: 50,

fontSize: 15,

text: 'Initial Field',

color: 'white'

},

// Specify the properties of the initial Dialog Settings in the intial field.

initialDialogSettings: {

displayMode: DisplayMode.Draw | DisplayMode.Upload | DisplayMode.Text,

hideSaveSignature: false

}

};

`

Textbox field settings

Using the updateFormField method, the form fields can be updated programmatically.

The following code example explains how to update the Textbox field properties on a button click.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="updateProperties">updateProperties</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Annotation,
Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Textbox',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Textbox',
thickness: 4,
value:'Textbox',
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
maxLength: 0,
isMultiline: false,
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="updateProperties">updateProperties</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Annotation,
Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Textbox',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Textbox',
thickness: 4,
value:'Textbox',
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
maxLength: 0,
isMultiline: false,
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }
});
}
}
}
</script>
```

The following code example explains how to update the properties of the textbox field added to the document from the form designer toolbar.

STANDALONE

```
<template>
```

```
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.textFieldSettings = {
// Set the name of the form field element.
name: 'Textbox',
// Specify whether the Textbox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the Textbox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Textbox',
// Set the thickness of the Textbox field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value:'Textbox',
// Set the font family of the textbox field.
fontFamily: 'Courier',
// Set the font size of the textbox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the textbox field.
```

```
color: 'black',
// Set the border color of the textbox field.
borderColor: 'black',
// Set the background color of the textbox field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,
// Allows multiline input in the text field. FALSE, by default.
isMultiline: false
}
},
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.textFieldSettings = {
// Set the name of the form field element.
```

```
name: 'Textbox',
// Specify whether the Textbox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the Textbox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Textbox',
// Set the thickness of the Textbox field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value:'Textbox',
// Set the font family of the textbox field.
fontFamily: 'Courier',
// Set the font size of the textbox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the textbox field.
color: 'black',
// Set the border color of the textbox field.
borderColor: 'black',
// Set the background color of the textbox field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,
// Allows multiline input in the text field. FALSE, by default.
isMultiline: false
}
},
}
}
</script>
```

Password field settings

Using the updateFormField method, the form fields can be updated programmatically.

The following code example explains how to update the Password field properties on a button click.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="updateProperties">updateProperties</button>
```

```
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Annotation,
Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Password',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Password',
thickness: 4,
value:'Password',
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
maxLength: 0,
bounds: { X: 148, Y: 229, Width: 150, Height: 24 }
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="updateProperties">updateProperties</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Annotation,
Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Password',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Password',
thickness: 4,
value:'Password',
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
maxLength: 0,
bounds: { X: 148, Y: 229, Width: 150, Height: 24 }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
});
}
}
}
</script>
```

The following code example explains how to update the properties of the password field added to the document from the form designer toolbar.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.passwordFieldSettings = {
// Set the name of the form field element.
name: 'Password',
// Specify whether the Password field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the Password field.
isPrint: true,
// Set the text to be displayed as a tooltip.
```

```
tooltip: 'Password',
// Set the thickness of the Password field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value:'Password',
// Set the font family of the Password field.
fontFamily: 'Courier',
// Set the font size of the Password field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the Password field.
color: 'black',
// Set the border color of the Password field.
borderColor: 'black',
// Set the background color of the Password field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,
}
},
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
};
```

```
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.passwordFieldSettings = {
// Set the name of the form field element.
name: 'Password',
// Specify whether the Password field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the Password field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Password',
// Set the thickness of the Password field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value:'Password',
// Set the font family of the Password field.
fontFamily: 'Courier',
// Set the font size of the Password field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the Password field.
color: 'black',
// Set the border color of the Password field.
borderColor: 'black',
// Set the background color of the Password field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,
}
},
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

CheckBox field settings

Using the updateFormField method, the form fields can be updated programmatically.

The following code example explains how to update the CheckBox field properties on a button click.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="updateProperties">updateProperties</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Annotation,
Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'CheckBox',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'CheckBox',
thickness: 4,
isChecked: true,
backgroundColor: '#daeaf7ff',
borderColor: 'black',
value: 'CheckBox'
});
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="updateProperties">updateProperties</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Annotation,
Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'CheckBox',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'CheckBox',
thickness: 4,
isChecked: true,
backgroundColor: '#daeaf7ff',
borderColor: 'black',
value: 'CheckBox'
});
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
}
</script>
```

The following code example explains how to update the properties of the checkbox field added to the document from the form designer toolbar.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.checkBoxFieldSettings = {
// Set the name of the form field element.
name: 'CheckBox',
// Specify whether the CheckBox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the CheckBox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'CheckBox',
```

```
// Set the thickness of the CheckBox field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Specifies whether the CheckBox is in checked state or not.
isSelected: true,
// Set the background color of the CheckBox  in hexadecimal string format.
backgroundColor: '#daeaf7ff',
// Set the border color of the CheckBox  field.
borderColor: 'black',
// Set the value of the form field element.
value: 'CheckBox'
}
},
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.checkBoxFieldSettings = {
// Set the name of the form field element.
```

Copyright © 2001 -2024 Syncfusion Inc.

```
name: 'CheckBox',
// Specify whether the CheckBox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the CheckBox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'CheckBox',
// Set the thickness of the CheckBox field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Specifies whether the CheckBox is in checked state or not.
isSelected: true,
// Set the background color of the CheckBox  in hexadecimal string format.
backgroundColor: '#daeaf7ff',
// Set the border color of the CheckBox  field.
borderColor: 'black',
// Set the value of the form field element.
value: 'CheckBox'
}
},
}
}
</script>
```

RadioButton field settings

Using the updateFormField method, the form fields can be updated programmatically.

The following code example explains how to update the RadioButton field properties on a button click.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="updateProperties">updateProperties</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
```

```
import { PdfViewerPlugin, Toolbar, Magnification, Annotation,
Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'RadioButton',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'RadioButton',
thickness: 4,
isSelected: true,
backgroundColor: '#daeaf7ff',
borderColor: 'black',
value: 'RadioButton'
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="updateProperties">updateProperties</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { PdfViewerPlugin, Toolbar, Magnification, Annotation,
Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'RadioButton',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'RadioButton',
thickness: 4,
isSelected: true,
backgroundColor: '#daeaf7ff',
borderColor: 'black',
value: 'RadioButton'
});
}
}
}
</script>
```

The following code example explains how to update the properties of the radiobutton field added to the document from the form designer toolbar.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.radioButtonFieldSettings = {
// Set the name of the form field element.
name: 'RadioButton',
// Specify whether the RadioButton field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the RadioButton field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'RadioButton',
// Set the thickness of the RadioButton field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Specifies whether the radio button is in checked state or not.
isSelected: true,
// Set the background color of the radio button in hexadecimal string
format.
backgroundColor: '#daeaf7ff',
// Set the border color of the radio button field.
borderColor: 'black',
// Set the value of the form field element.
value: 'RadioButton'
}
},
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.checkBoxFieldSettings = {
viewer.radioButtonFieldSettings = {
// Set the name of the form field element.
name: 'RadioButton',
// Specify whether the RadioButton field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the RadioButton field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'RadioButton',
// Set the thickness of the RadioButton field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Specifies whether the radio button is in checked state or not.
isSelected: true,
// Set the background color of the radio button in hexadecimal string
format.
backgroundColor: '#daeaf7ff',
```

```
// Set the border color of the radio button field.
borderColor: 'black',
// Set the value of the form field element.
value: 'RadioButton'
}
},
}
}
</script>
```

ListBox field settings

Using the updateFormField method, the form fields can be updated programmatically.

The following code example explains how to update the ListBox field properties on a button click.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="updateProperties">updateProperties</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Annotation,
Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties() {
```

```
viewer = this.$refs.pdfviewer.ej2Instances;
const formFields = viewer.retrieveFormFields();
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
viewer.formDesignerModule.updateFormField(formFields[0], {
name: 'ListBox',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'ListBox',
thickness: 4,
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
options: customOptions
});
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="updateProperties">updateProperties</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Annotation,
Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties() {
viewer = this.$refs.pdfviewer.ej2Instances;
const formFields = viewer.retrieveFormFields();
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
viewer.formDesignerModule.updateFormField(formFields[0], {
name: 'ListBox',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'ListBox',
thickness: 4,
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
options: customOptions
});
}
}
}
</script>
```

The following code example explains how to update the properties of the listbox field added to the document from the form designer toolbar.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
viewer.listBoxFieldSettings = {
// Set the name of the form field element.
name: 'listbox',
// Specify whether the listbox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the listbox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'listbox',
// Set the thickness of the listbox field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value:'listbox',
// Set the font family of the listbox field.
fontFamily: 'Courier',
// Set the font size of the listbox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the  font color of the listbox field.
color: 'black',
// Set the border color of the listbox field.
borderColor: 'black',
// Set the background color of the listbox field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
```

```
alignment: 'Left',
// Set the listbox items.
options: customOptions
}
},
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
viewer.listBoxFieldSettings = {
// Set the name of the form field element.
name: 'listbox',
// Specify whether the listbox field is in read-only or read-write mode.
isReadOnly: false,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the listbox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'listbox',
// Set the thickness of the listbox field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value:'listbox',
// Set the font family of the listbox field.
fontFamily: 'Courier',
// Set the font size of the listbox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the  font color of the listbox field.
color: 'black',
// Set the border color of the listbox field.
borderColor: 'black',
// Set the background color of the listbox field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the listbox items.
options: customOptions
}
},
}
}
</script>
```

DropDown field settings

Using the updateFormField method, the form fields can be updated programmatically.

The following code example explains how to update the DropDown field properties on a button click.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="updateProperties">updateProperties</button>
<ejs-pdfviewer
id="pdfViewer"
```

Copyright © 2001 -2024 Syncfusion Inc.

```
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Annotation,
Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties() {
viewer = this.$refs.pdfviewer.ej2Instances;
const formFields = viewer.retrieveFormFields();
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
viewer.formDesignerModule.updateFormField(formFields[0], {
name: 'DropDown',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'DropDown',
thickness: 4,
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor:'#daeaf7ff',
alignment: 'Left',
options: customOptions,
});
}
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="updateProperties">updateProperties</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Annotation,
Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Event triggers on the Update Properties button click.
updateProperties() {
viewer = this.$refs.pdfviewer.ej2Instances;
const formFields = viewer.retrieveFormFields();
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
viewer.formDesignerModule.updateFormField(formFields[0], {
name: 'DropDown',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'DropDown',
thickness: 4,
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
color: 'black',
borderColor: 'black',
backgroundColor:'#daeaf7ff',
alignment: 'Left',
options: customOptions,
});
}
}
}
</script>
```

The following code example explains how to update the properties of the dropdown field added to the document from the form designer toolbar.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
viewer.DropdownFieldSettings = {
// Set the name of the form field element.
```

Copyright © 2001 -2024 Syncfusion Inc.

```
name: 'DropDown',
// Specify whether the DropDown field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the DropDown field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'DropDown',
// Set the thickness of the DropDown field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value:'DropDown',
// Set the font family of the DropDown field.
fontFamily: 'Courier',
// Set the font size of the DropDown field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the  font color of the DropDown field.
color: 'black',
// Set the border color of the DropDown field.
borderColor: 'black',
// Set the background color of the DropDown field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the DropDown items.
options: customOptions
}
},
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
viewer.DropdownFieldSettings = {
// Set the name of the form field element.
name: 'DropDown',
// Specify whether the DropDown field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the DropDown field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'DropDown',
// Set the thickness of the DropDown field. To hide the borders, set the
value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value:'DropDown',
// Set the font family of the DropDown field.
fontFamily: 'Courier',
// Set the font size of the DropDown field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the  font color of the DropDown field.
color: 'black',
// Set the border color of the DropDown field.
borderColor: 'black',
// Set the background color of the DropDown field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
```

```
// Set the DropDown items.
options: customOptions
}
},
}
}
</script>
```

Create form fields with UI interaction in Vue Pdfviewer component

The PDF viewer control provides the option for interaction with Form Fields such as Drag and resize. you can draw a Form Field dynamically by clicking the Form Field icon on the toolbar and draw it in the PDF document. The Form Fields type supported by the PDF Viewer Control are:

- Textbox
- Password
- CheckBox
- RadioButton
- ListBox
- DropDown
- SignatureField
- InitialField

Enable or Disable form designer toolbar

We should inject FormDesigner module and set enableFormDesignerToolbar as true to enable the Form designer icon on the toolbar. By default, enableFormDesignerToolbar is set as true. Use the following code to inject FormDesigner module and to enable the enableFormDesignerToolbar property.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:enableFormDesignerToolbar= "false">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
```

```
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]}
}
</script>
```

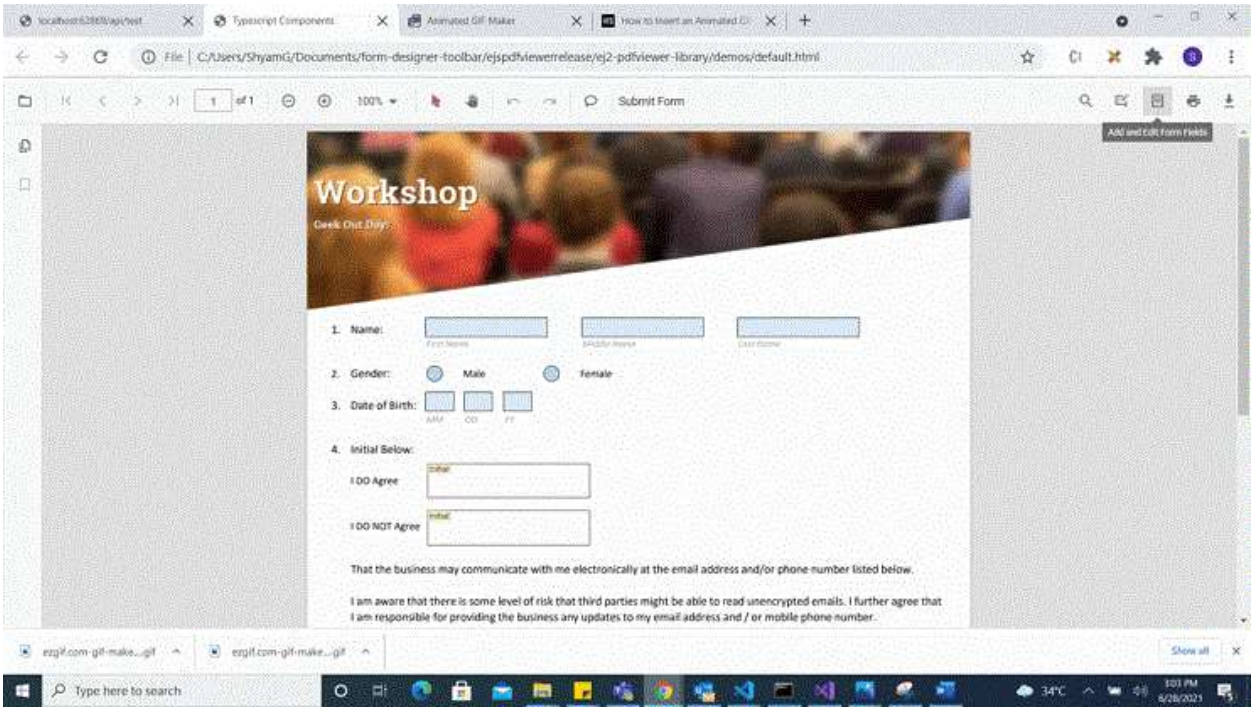
SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl"
:enableFormDesignerToolbar= "false">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]}
}
</script>
```

Add the form field dynamically

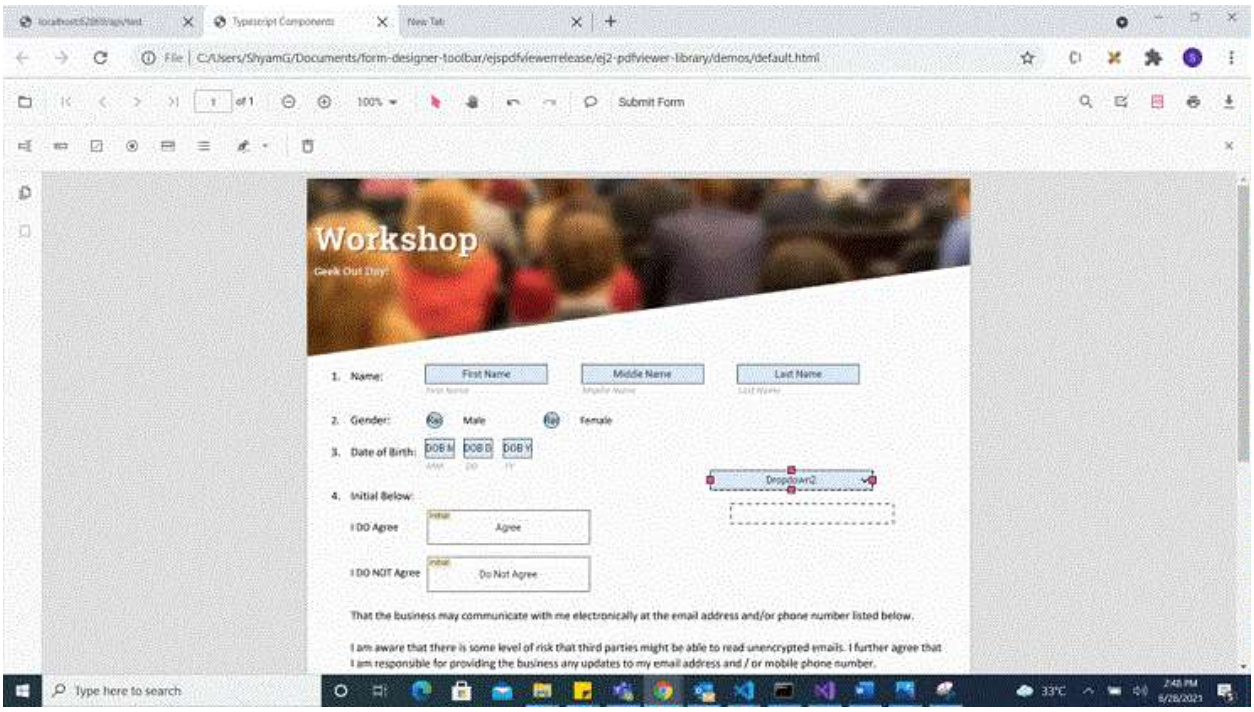
Click the Form Field icon on the toolbar and then click on to the PDF document to draw a Form Field. Refer the below GIF for further reference.

Copyright © 2001 -2024 Syncfusion Inc.



Drag the form field

We provide options to drag the Form Field which is currently selected in the PDF document. Refer the below GIF for further reference.



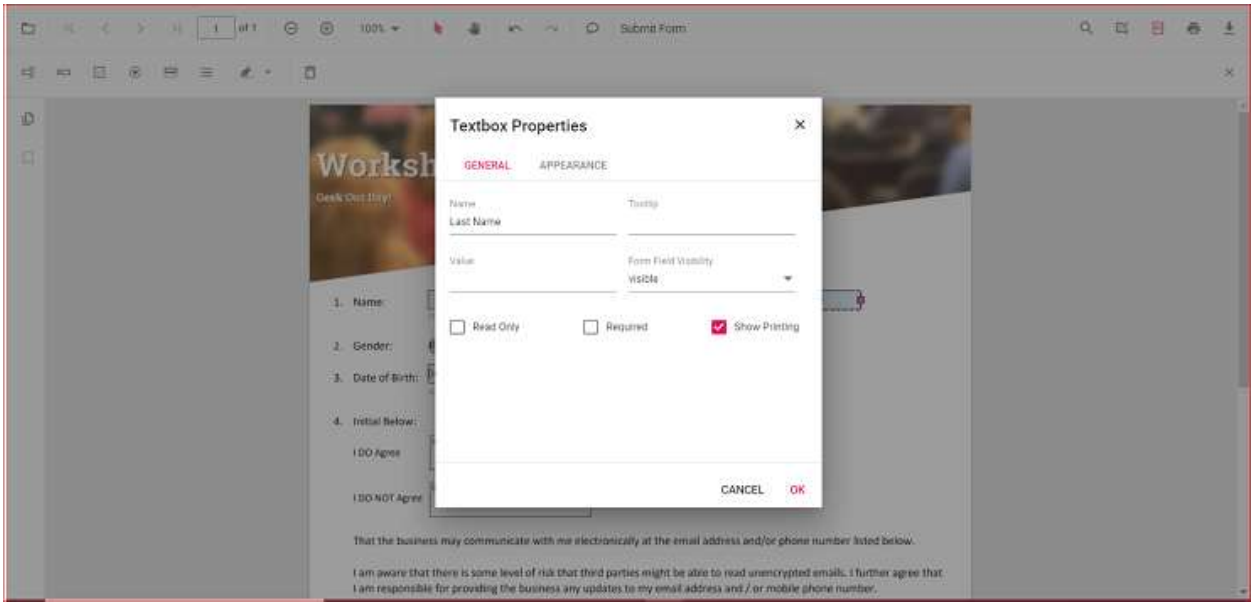
Resize the form field

We provide options to resize the Form Field which is currently selected in the PDF document. Refer the below GIF for further reference.

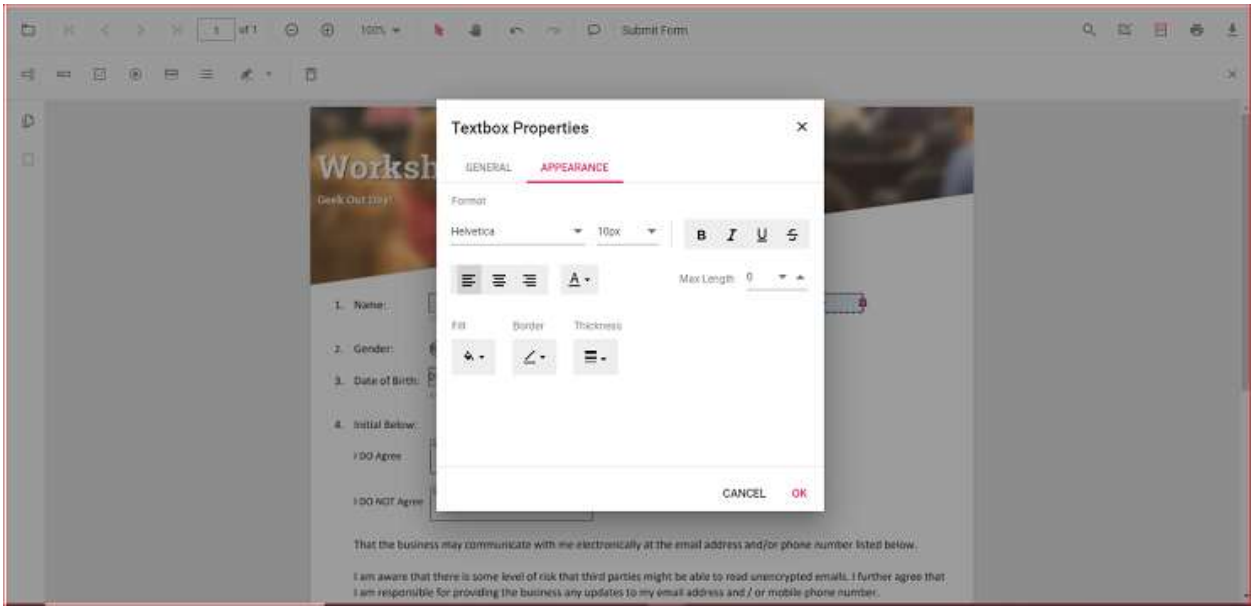
Copyright © 2001 -2024 Syncfusion Inc.

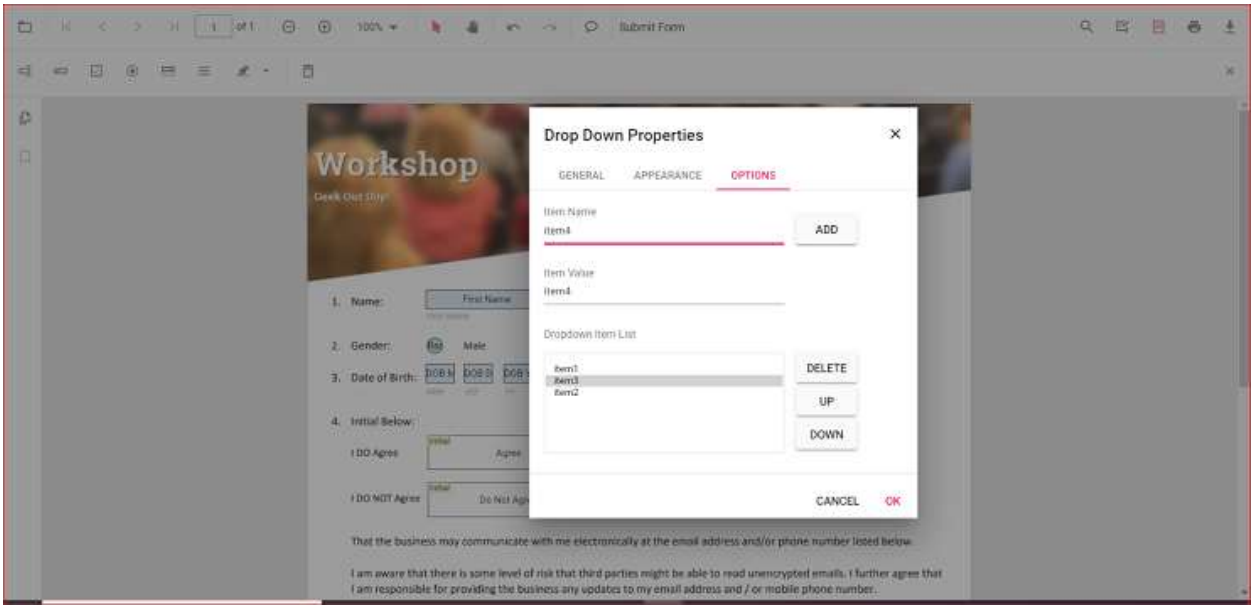
Edit or Update the form field dynamically

The properties of the Form Fields can be edited using the Form Field Properties window. It can be opened by selecting the Properties option in the context menu that appears on the right by clicking the Form Field object. Refer the below image for the properties available to customize the appearance of the Form Field.



Copyright © 2001 -2024 Syncfusion Inc.

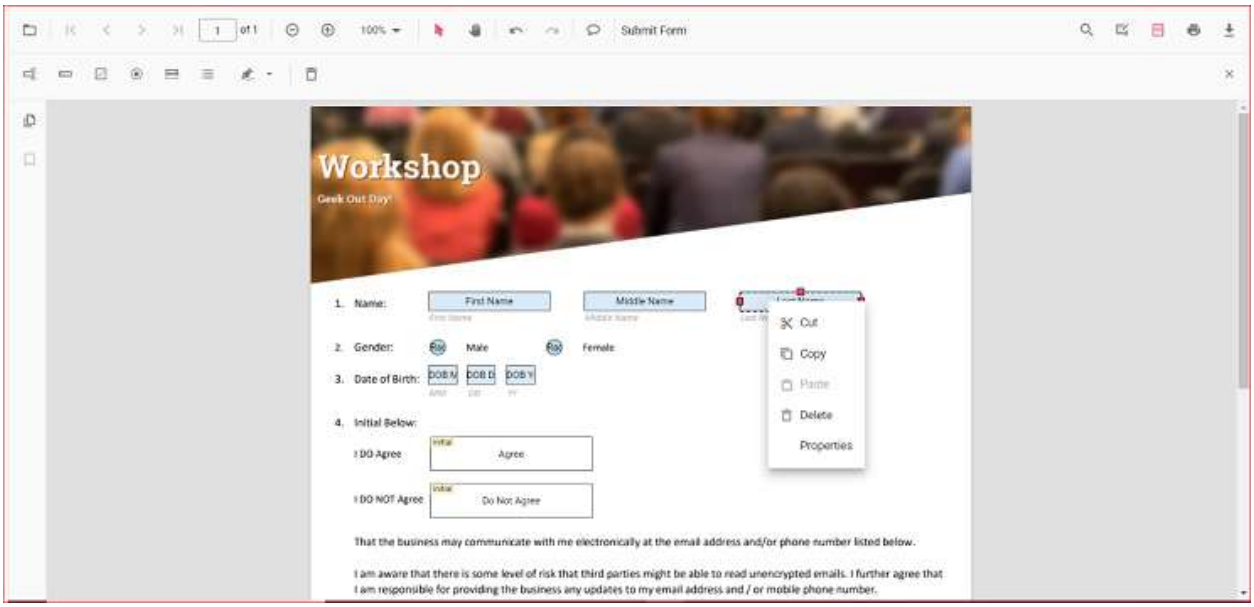




Clipboard operation with form field

The PDF Viewer control supports the clipboard operations such as cut, copy and paste for Form Fields. You can right click on the Form Field object to view the context menu and select to the clipboard options that you would like to perform. Refer the below image for the options in the context menu.

Copyright © 2001 -2024 Syncfusion Inc.



Undo and Redo

We provided support to undo/redo the Form Field actions that are performed at runtime. Use the following code example to perform undo/redo actions.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="undoClicked">Undo</button>
<button v-on:click="redoClicked">Redo</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
methods: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
undoClicked: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.undo();
},
redoClicked: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.redo();
},
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="undoClicked">Undo</button>
<button v-on:click="redoClicked">Redo</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/form-designer.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
methods: {
undoClicked: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.undo();
},
redoClicked: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
viewer.redo();
},
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
```

Print in Vue Pdfviewer component

The PDF Viewer supports printing the loaded PDF file. You can enable/disable the print using the following code snippet.

`

<!DOCTYPE html>

<html lang="en">

<head>

<title>Essential JS 2</title>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no" />

<meta name="description" content="Essential JS 2" />

<meta name="author" content="Syncfusion" />

<link rel="shortcut icon" href="resources/favicon.ico" />

<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet" />

<!--style reference from app-->

<link href="/styles/styles.css" rel="stylesheet" />

<!--system js reference and configuration-->

<script src="node_modules/systemjs/dist/system.src.js" type="text/javascript"></script>

<script src="system.config.js" type="text/javascript"></script>

</head>

<body>

<!--Element which will render as PdfViewer -->

</body>

</html>

`

`ts

import { PdfViewer, Toolbar, Magnification, Navigation, LinkAnnotation, Annotation, ThumbnailView,BookmarkView, TextSelection} from '@syncfusion/ej2-pdfviewer';

PdfViewer.Inject(Toolbar,Magnification,Navigation, LinkAnnotation, Annotation, ThumbnailView,BookmarkView, TextSelection);

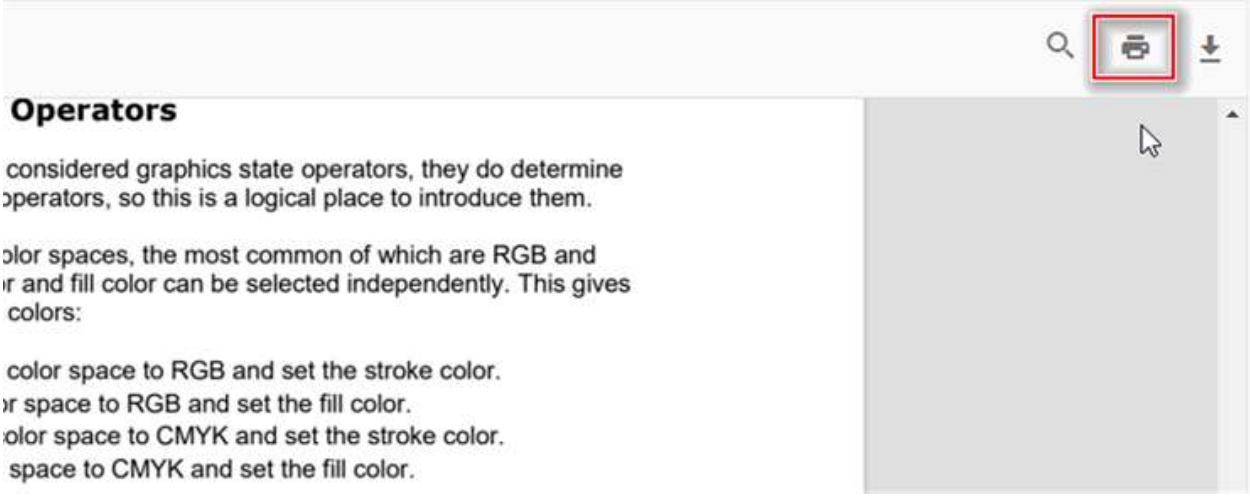
Copyright © 2001 -2024 Syncfusion Inc.

let pdfviewer: PdfViewer = new PdfViewer({enablePrint: true, documentPath:'PDF_Succinctly.pdf'});

pdfviewer.serviceUrl = 'https://services.syncfusion.com/vue/production/api/pdfviewer';

pdfviewer.appendTo('#PdfViewer');

`



You can invoke print action using the following code snippet.,

STANDALONE

```
<template>
<div id="app">
<button v-on:click="printClicked">Print</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
printClicked: function (args) {
viewer.print.print();
},
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
},
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="printClicked">Print</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
printClicked: function (args) {
viewer.print.print();
},
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
},
}
```

Copyright © 2001 -2024 Syncfusion Inc.

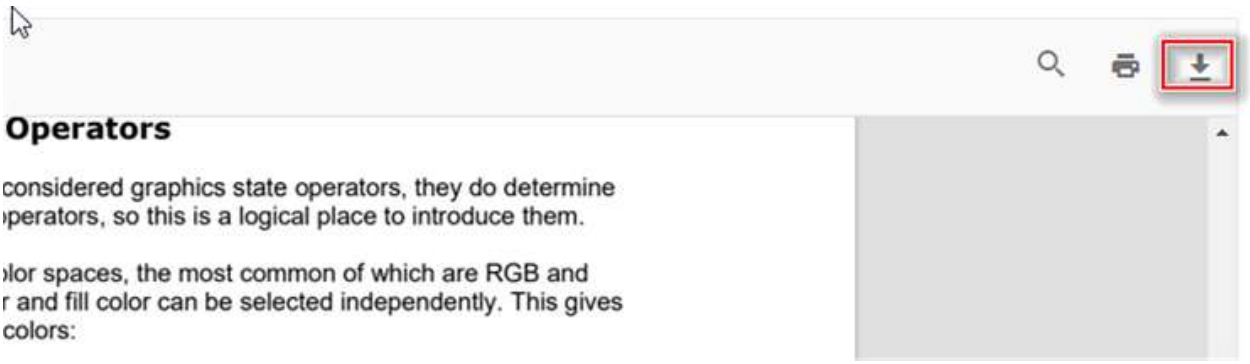
```
}
</script>
```

See also

- Toolbar items
- Feature Modules

Download in Vue Pdfviewer component

The PDF Viewer supports downloading the loaded PDF file. You can enable/disable the download using the following code snippet.



You can invoke download action using following code snippet.,

STANDALONE

```
<template>
<div id="app">
<button v-on:click="downloadClicked">Download</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
```

```
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
downloadClicked: function (args) {
viewer.download();
},
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
},
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="downloadClicked">Download</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
downloadClicked: function (args) {
viewer.download();
},
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
}
}
</script>
```

See also

- Toolbar items
- Feature Modules

Globalization in Vue Pdfviewer component

The text contents provided in the PDF Viewer can be localized using the collection of localized strings for different cultures. By default, the PDF Viewer is localized in "**en-US**".

The following table shows the default text values used in PDF Viewer in 'en-US' culture:

|Keywords|Values|

|---|---|

|PdfViewer|PDF Viewer|

|Cancel|Cancel|

|Download file|Download file|

|Download|Download|

|Enter Password|This document is password protected. Please enter a password.|

|File Corrupted|File corrupted|

|File Corrupted Content|The file is corrupted and cannot be opened.|

|Fit Page|Fit page|

|Fit Width|Fit width|

|Automatic|Automatic|

|Go To First Page|Show first page|

|Invalid Password|Incorrect password. Please try again.|

|Next Page|Show next page|

|OK|OK|

|Open|Open file|

|Page Number|Current page number|

|Previous Page|Show previous page|

|Go To Last Page|Show last page|

|Zoom|Zoom|

|Zoom In|Zoom in|

|Zoom Out|Zoom out|

Copyright © 2001 -2024 Syncfusion Inc.

|Page Thumbnails|Page thumbnails|

|Bookmarks|Bookmarks|

|Print|Print file

|Password Protected|Password required|

|Copy|Copy|

|Text Selection|Text selection tool|

|Panning|Pan mode|

|Text Search|Find text|

|Find in document|Find in document|

|Match case|Match case|

|Apply|Apply|

|GoToPage|Go to page|

|No matches|Viewer has finished searching the document. No more matches were found|

|No Text Found|No Text Found|

|Undo|Undo|

|Redo|Redo|

|Annotation|Add or Edit annotations|

|Highlight|Highlight Text|

|Underline|Underline Text|

|Strikethrough|Strikethrough Text|

|Delete|Delete annotation|

|Opacity|Opacity|

|Color edit|Change Color|

|Opacity edit|Change Opacity|

|Highlight context|Highlight|

|Underline context|Underline|

|Strikethrough context|Strike through|

|Server error|Web-service is not listening. PDF Viewer depends on web-service for all it's features. Please start the web service to continue.|

|Open text|Open|

|First text|First Page|

|Previous text|Previous Page|

|Next text|Next Page|

Copyright © 2001 -2024 Syncfusion Inc.

|Last text|Last Page|

|Zoom in text|Zoom In|

|Zoom out text|Zoom Out|

|Selection text|Selection|

|Pan text|Pan|

|Print text|Print|

|Search text|Search|

|Annotation Edit text|Edit Annotation| Please start the web service to continue.|

The different locale value for the PDF Viewer can be specified using the locale property.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
locale='ar-AE'>
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from "vue";
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
Annotation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner} from
'@syncfusion/ej2-vue-pdfviewer';
import { L10n, setCulture  } from '@syncfusion/ej2-base';
Vue.use(PdfViewerPlugin);
export default Vue.extend({
data: function() {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
}
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner ]}
});
//PDF Viewer Arabic Sample Locale
L10n.load({
'ar-AE': {
'PdfViewer' : {
'PdfViewer': 'قوات الدفاع الشعبي المشاهد',
'Cancel' : 'إلغاء',
'Download file' : 'تحميل الملف',
'Download': 'تحميل',
'Enter Password' : 'يرجى إدخال كلمة مرور. هذا المستند محمي بكلمة مرور.',
'File Corrupted' : 'ملف تالف',
'File Corrupted Content' : 'الملف تالف ولا يمكن فتحه.',
```

```
'Fit Page' : 'لائق بدنيا الصفحة',
'Fit Width' : 'لائق بدنيا عرض',
'Automatic' : 'تلقائي',
'Go To First Page' : 'عرض الصفحة الأولى',
'Invalid Password' : 'كلمة سر خاطئة. حاول مرة اخرى.',
'Next Page' : 'عرض الصفحة التالية',
'OK': 'حسنا',
'Open' : 'فتح الملف',
'Page Number' : 'رقم الصفحة الحالية',
'Previous Page' : 'عرض الصفحة السابقة',
'Go To Last Page' : 'عرض الصفحة الأخيرة',
'Zoom' : 'تكبير',
'Zoom In' : 'تكبير في',
'Zoom Out' : 'تكبير خارج',
'Page Thumbnails': 'مصغرات الصفحة',
'Bookmarks': 'المرجعية',
'Print' : 'اطبع الملف',
'Password Protected' : 'كلمة المرور مطلوبة',
'Copy': 'نسخ',
'Text Selection': 'أداة اختيار النص',
'Panning': 'وضع عموم',
'Text Search': 'بحث عن نص',
'Find in document': 'ابحث في المستند',
'Match case': 'حالة مباراة',
'Apply': 'تطبيق',
'GoToPage': 'انتقل إلى صفحة',
// tslint:disable-next-line:max-line-length
'No matches': 'انتهى العارض من البحث في المستند. لم يتم العثور على مزيد من
التطابقات',
'No Text Found': 'لم يتم العثور على نص',
// tslint:disable-next-line:max-line-length
'Server error': 'خدمة الانترنت لا يستمع. يعتمد قوات الدفاع الشعبي المشاهد على
خدمة الويب لجميع ميزاته. يرجى بدء خدمة الويب للمتابعة.',
'Undo' : 'فك',
'Redo' : 'فعل ثانية',
'Annotation': 'إضافة أو تعديل التعليقات التوضيحية',
'Highlight': 'تسليط الضوء على النص',
'Underline': 'تسطير النص',
'Strikethrough': 'نص يتوسطه خط',
'Delete': 'حذف التعليق التوضيحي',
'Opacity': 'غموض',
'Color edit': 'غير اللون',
'Opacity edit': 'تغيير التعتيم',
'Highlight context': 'تسليط الضوء',
'Underline context': 'أكد',
'Strikethrough context': 'يتوسطه',
'Open text': 'افتح',
'First text': 'الصفحة الأولى',
'Previous text': 'الصفحة السابقة',
'Next text': 'الصفحة التالية',
'Last text': 'آخر صفحة',
'Zoom in text': 'تكبير',
'Zoom out text': 'تصغير',
'Selection text': 'اختيار',
'Pan text': 'مقلاة',
'Print text': 'طباعة',
'Search text': 'بحث',
```

```
'Annotation Edit text': 'تحرير التعليق التوضيحي'
}
}
});
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl"
locale='ar-AE'>
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from "vue";
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
Annotation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner} from
'@syncfusion/ej2-vue-pdfviewer';
import { L10n, setCulture  } from '@syncfusion/ej2-base';
Vue.use(PdfViewerPlugin);
export default Vue.extend({
data: function() {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
}
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, Annotation,
BookmarkView,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner ]}
});
//PDF Viewer Arabic Sample Locale
L10n.load({
'ar-AE': {
'PdfViewer' : {
'PdfViewer': 'قوات الدفاع الشعبي المشاهد',
'Cancel' : 'إلغاء',
'Download file' : 'تحميل الملف',
'Download': 'تحميل',
'Enter Password' : 'يرجى إدخال كلمة مرور. هذا المستند محمي بكلمة مرور.',
'File Corrupted' : 'ملف تالف',
'File Corrupted Content' : 'الملف تالف ولا يمكن فتحه.',
'Fit Page' : 'لائق بدنيا الصفحة',
'Fit Width' : 'لائق بدنيا عرض',
'Automatic' : 'تلقائي',
'Go To First Page' : 'عرض الصفحة الأولى',
'Invalid Password' : 'كلمة سر خاطئة. حاول مرة اخرى.',
'Next Page' : 'عرض الصفحة التالية',
'OK': 'حسنا',
```

```
'Open' : 'فتح الملف',
'Page Number' : 'رقم الصفحة الحالية',
'Previous Page' : 'عرض الصفحة السابقة',
'Go To Last Page' : 'عرض الصفحة الأخيرة',
'Zoom' : 'تكبير',
'Zoom In' : 'تكبير في',
'Zoom Out' : 'تكبير خارج',
'Page Thumbnails': 'مصغرات الصفحة',
'Bookmarks': 'المرجعية',
'Print' : 'اطبع الملف',
'Password Protected' : 'كلمة المرور مطلوبة',
'Copy': 'نسخ',
'Text Selection': 'أداة اختيار النص',
'Panning': 'وضع عموم',
'Text Search': 'بحث عن نص',
'Find in document': 'ابحث في المستند',
'Match case': 'حالة مباراة',
'Apply': 'تطبيق',
'GoToPage': 'انتقل إلى صفحة',
// tslint:disable-next-line:max-line-length
'No matches': 'انتهى العارض من البحث في المستند. لم يتم العثور على مزيد من التطابقات',
'No Text Found': 'لم يتم العثور على نص',
// tslint:disable-next-line:max-line-length
'Server error': 'خدمة الانترنت لا يستمع. يعتمد قوات الدفاع الشعبي المشاهد على خدمة الويب لجميع ميزاته. يرجى بدء خدمة الويب للمتابعة.',
'Undo' : 'فك',
'Redo' : 'فعل ثانية',
'Annotation': 'إضافة أو تعديل التعليقات التوضيحية',
'Highlight': 'تسليط الضوء على النص',
'Underline': 'تسطير النص',
'Strikethrough': 'نص يتوسطه خط',
'Delete': 'حذف التعليق التوضيحي',
'Opacity': 'غموض',
'Color edit': 'غير اللون',
'Opacity edit': 'تغيير التعتيم',
'Highlight context': 'تسليط الضوء',
'Underline context': 'أكد',
'Strikethrough context': 'يتوسطه',
'Open text': 'افتح',
'First text': 'الصفحة الأولى',
'Previous text': 'الصفحة السابقة',
'Next text': 'الصفحة التالية',
'Last text': 'آخر صفحة',
'Zoom in text': 'تكبير',
'Zoom out text': 'تصغير',
'Selection text': 'اختيار',
'Pan text': 'مقلاة',
'Print text': 'طباعة',
'Search text': 'بحث',
'Annotation Edit text': 'تحرير التعليق التوضيحي'
}
}
});
</script>
```

How To

Toolbar customization in Vue Pdfviewer component

The PDF Viewer provides API for user interactions options provided in it's built-in toolbar. Using this we can create our own User Interface for toolbar actions in application level by hiding the default toolbar. The following steps are used to create the custom toolbar for PDF Viewer,

Step 1: Follow the steps provided in the link to create simple PDF Viewer sample.

Step 2: Now, add an HTML div element in template to act as the custom toolbar PDF Viewer using the following code.

STANDALONE

```
<template>
<div>
<ejs-toolbar id="customToolbar" ref="toolbar">
<e-items>
<e-item prefixIcon='e-pv-open-document-icon' tooltipText='Open'
:click="openClicked"></e-item>
<e-item prefixIcon='e-pv-previous-page-navigation-icon' id="previousPage"
tooltipText='Previous Page' align='Center' :click="previousClicked"></e-
item>
<e-item prefixIcon='e-pv-next-page-navigation-icon' id="nextPage"
tooltipText='Next Page' align='Center' :click="nextClicked"></e-item>
<e-item :template='pageNoTemplate' tooltipText='Page Number'
align='Center'></e-item>
<e-item :template='pageTextTemplate' tooltipText='Page Number'
align='Center'></e-item>
<e-item prefixIcon='e-pv-print-document-icon' tooltipText='Print'
align='Right' :click="printClicked"></e-item>
<e-item prefixIcon='e-pv-download-document-icon' tooltipText='Download'
align='Right' :click="downloadClicked"></e-item>
</e-items>
</ejs-toolbar>
<ejs-toolbar id="magnificationToolbar">
<e-items>
<e-item prefixIcon='e-pv-fit-page-icon' id="fitPage" tooltipText='Fit to
page' :click="pageFitClicked"></e-item>
<e-item prefixIcon='e-pv-zoom-in-icon' id="zoomIn" tooltipText='Zoom in'
:click="zoomInClicked"></e-item>
<e-item prefixIcon='e-pv-zoom-out-icon' id="zoomOut" tooltipText='Zoom out'
:click="zoomOutClicked"></e-item>
</e-items>
</ejs-toolbar>
<input type ="file" id="fileUpload" accept=".pdf"
style="display:block;visibility:hidden;width:0;height:0;"></input>
<div class="control-section">
<ejs-pdfviewer
id="pdfviewer"
ref="pdfviewer"
:pageChange="pageChange"
:documentLoad="documentLoad"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:enableToolbar="enableToolbar">
</ejs-pdfviewer>
```

```
</div>
</div>
</template>
```

SERVER-BACKED

```
<template>
<div>
<ejs-toolbar id="customToolbar" ref="toolbar">
<e-items>
<e-item prefixIcon='e-pv-open-document-icon' tooltipText='Open'
:click="openClicked"></e-item>
<e-item prefixIcon='e-pv-previous-page-navigation-icon' id="previousPage"
tooltipText='Previous Page' align='Center' :click="previousClicked"></e-
item>
<e-item prefixIcon='e-pv-next-page-navigation-icon' id="nextPage"
tooltipText='Next Page' align='Center' :click="nextClicked"></e-item>
<e-item :template='pageNoTemplate' tooltipText='Page Number'
align='Center'></e-item>
<e-item :template='pageTextTemplate' tooltipText='Page Number'
align='Center'></e-item>
<e-item prefixIcon='e-pv-print-document-icon' tooltipText='Print'
align='Right' :click="printClicked"></e-item>
<e-item prefixIcon='e-pv-download-document-icon' tooltipText='Download'
align='Right' :click="downloadClicked"></e-item>
</e-items>
</ejs-toolbar>
<ejs-toolbar id="magnificationToolbar">
<e-items>
<e-item prefixIcon='e-pv-fit-page-icon' id="fitPage" tooltipText='Fit to
page' :click="pageFitClicked"></e-item>
<e-item prefixIcon='e-pv-zoom-in-icon' id="zoomIn" tooltipText='Zoom in'
:click="zoomInClicked"></e-item>
<e-item prefixIcon='e-pv-zoom-out-icon' id="zoomOut" tooltipText='Zoom out'
:click="zoomOutClicked"></e-item>
</e-items>
</ejs-toolbar>
<input type ="file" id="fileUpload" accept=".pdf"
style="display:block;visibility:hidden;width:0;height:0;"></input>
<div class="control-section">
<ejs-pdfviewer
id="pdfviewer"
ref="pdfviewer"
:pageChange="pageChange"
:documentLoad="documentLoad"
:documentPath="documentPath"
:enableToolbar="enableToolbar">
</ejs-pdfviewer>
</div>
</div>
</template>
```

Step 3: Import and inject the modules used for the custom toolbar,

`ts

import Vue from "vue";

import { PdfViewerPlugin, Toolbar, Magnification, Navigation,

LinkAnnotation, BookmarkView, Annotation, ThumbnailView,

Print, TextSelection, TextSearch, FormFields, FormDesigner } from "@syncfusion/ej2-vue-pdfviewer";

import { ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";

import { CalendarPlugin } from '@syncfusion/ej2-vue-calendars';

Vue.use(CalendarPlugin);

Vue.use(PdfViewerPlugin);

Vue.use(ToolbarPlugin);

`

Step 4: Hide the default toolbar of PDF Viewer using below code snippet,

STANDALONE

```
export default Vue.extend({
data: function () {
return {
documentPath: "https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf",
enableToolbar: false,
pageNoTemplate: '<div class=""><input type="text" class="e-input-group e-pv-current-page-number" id="currentPage" /></div>',
pageTextTemplate: '<div class=""><span class="e-pv-total-page-number" id="totalPage">of 0</span></div>'
}
})
};
```

SERVER-BACKED

```
export default Vue.extend({
data: function () {
return {
serviceUrl: "https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath: "https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf",
enableToolbar: false,
pageNoTemplate: '<div class=""><input type="text" class="e-input-group e-pv-current-page-number" id="currentPage" /></div>',
pageTextTemplate: '<div class=""><span class="e-pv-total-page-number" id="totalPage">of 0</span></div>'
}
})
};
```

Step 7: Add the following style to achieve the custom toolbar styling,

`

<style>

Copyright © 2001 -2024 Syncfusion Inc.

```
pdfviewer {
height: 640px;
}
.control-section {
padding-top: 0px !important;
}
magnificationToolbar {
background: transparent;
height: auto;
min-height: 56px;
width: auto;
border: none;
position: absolute;
z-index: 900;
top: calc(100% - 110px);
left: calc(100% - 120px);
transform: rotate(90deg);
}
div#magnificationToolbar.e-toolbar .e-toolbar-items {
background: transparent;
padding: 2px 3px 2px 2px;
}
magnificationToolbar.e-toolbar .e-tbar-btn {
border-radius: 50%;
min-height: 30px;
min-width: 30px;
border: 1px solid #c8c8c8;
}
customToolbar {
top: 0px;
z-index: 900;
}
.e-bookmark-popup {
height: 300px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
max-width: 300px;
}
.e-text-search-popup {
height: 104px;
max-width: 348px;
}
.e-custom-search-input {
width: 234px;
}
.e-text-search-popup .e-footer-content, .e-bookmark-popup .e-footer-content {
padding: 0;
height: 0;
}
.search-button, .search-button:disabled, .search-button:focus, .search-button:hover {
background: transparent;
box-shadow: none;
}
popup .e-dlg-content {
padding-left: 0;
padding-bottom: 0;
}
.e-pv-bookmarks {
min-width: 234px;
}
.e-pv-current-page-number {
width: 46px;
height: 28px;
text-align: center;
}
.material .e-pv-current-page-number {
border-width: 1px;
}
.e-icons {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
font-family: "e-icons";
font-style: normal;
font-variant: normal;
font-weight: normal;
line-height: 1;
text-transform: none;
}
.e-pv-icon::before {
font-family: 'e-icons';
}
.e-pv-open-document-icon::before  {
content: '\e91c';
}
.e-pv-download-document-icon::before {
content: '\e914';
}
.e-pv-print-document-icon::before {
content: '\e917';
}
.e-pv-previous-page-navigation-icon::before {
content: '\e910';
}
.e-pv-next-page-navigation-icon::before {
content: '\e911';
}
.e-pv-zoom-out-icon::before {
content: '\e912';
}
.e-pv-zoom-in-icon::before {
content: '\e91d';
}
.e-pv-fit-page-icon::before {
content: '\e91b';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
.e-btn-icon.e-pv-zoom-out-icon.e-icons {
transform: rotate(90deg);
}
@font-face {
font-family: "e-icons";
font-style: normal;
font-weight: normal;
src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj8wS0QAAAEoAAAAVmNtYXDSeNLMAAABuAAAAFZnbHlmok0Nt
wAAAjAAAAPkaGVhZBN3pEcAAADQAAAANmhoZWEHrwNhAAAArAAAACRobXR4NsgAAAAAAYAAAAA4b
G9jYQdkBmQAAAIQAAAAHm1heHABHAAwAAABCAAAACBuYW1lD0oZXgAABhQAAALBcG9zdFG4mE4AA
AjYAAAAyAABAAADUv9qAFoEAAAA/+gEAAABAAAAAAAAAAAAAAAAAAAAADgABAAAAAQAAxsly1F8PPPU
ACwPoAAAAANgsr7EAAAAA2CyvsQAAAAAEAAQAAAAACAACAAAAAAAAAAEAAAAOACQABAAAAAAAAg
AAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA6RDpHQNS/2oAWgQAAJYAAAABAAAAAAAABAAAAAPoAAAD6
AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAAAACAAAAAwAA
ABQAAwABAAAAFAAEAEIAAAAGAAQAAQAC6RLpHf//AADpEOkU//8AAAAAAAEABgAKAAAAAQACAAMA
BQAGAAcACAAJAAoACwAMAA0ABAAAAAAAAAAAUACoAZACkAL4A7gEuAVwBcAGEAZ4ByAHyAAAAAQAA
AAAD6gMuAAUAAAkBBwkBJwIAAet0/on+iXQDL/4VcwF3/olzAAEAAAAAAA+oDLgAFAAATCQEXCQGJAXcB
d3T+Ff4VAy/+iQF3c/4VAesAAAAAAwAAAAAEAAQAAAMADwAbAAABITUhBQ4BBy4BJz4BNx4BBRYAFzYA
NyYAJwYAAQACAP4AAoAE2aOj2QQE2aOj2fyEBgEh2dkBIQYG/t/Z2f7fAcCAQKPZBATZo6PZBATZo9n+3wY
GASHZ2QEhBgb+3wAAAADAAAAAAQABAAACwAXACMAAAEjFTMVMzUzNSM1IwEOAQcuASc+ATceAQ
UWABc2ADcmACcGAAHAwMCAwMCAAcAE2aOj2QQE2aOj2fyEBgEh2dkBIQYG/t/Z2f7fAkCAwMCAwP8A
o9kEBNmjo9kEBNmj2f7fBgYBIdnZASEGBv7fAAIAAAAAAwAEAAADAAoAADEhNSEBIQkBIREhAwD9AAEA/
wABgAGA/wD/AIACAP6AAYABgAACAAAAAANABAAADgAaAAABMh4CFRElBRE0Nz4BMycGFRElBRE0JiMh
IgKdCwwHBf7g/uAJBAwKdC8BoAGgX0T+BkQDgAYGCwr9YHZ2AqAOCQQGUS9D/KGrqwNfRlsAAAACAAA
AAAP/BAAACwAjAAABDgEHLgEnPgE3HgEFHgEXMjY/ARcVATcBIyc3PgE1LgEnDgECgAOQbW2QAwOQbW
2Q/YME2aNGfDIDJAEEYf78MyMCKi4E2aOj2QKAbZADA5BtbZADA5Bto9kELioDJDP+/GEBBCQDMnxGo9k
EBNkAAAQAAAAABAAEAAADAAcAFQAZAAABFSE1JRUjNSERMxUhNTMRLgEnIQ4BNyE1IQLA/oACQID9A
MACgMABSDf9ADdIvwKA/YABwMDAwICA/sDAwAFAN0gBAUmKwAAAAQAAAAACQAQAAAUAABEBNwk
BJwHsU/6HAXpSAmD+YGIBPgE+YgAAAAEAAAAAAkAEAAAFAAARCQEXCQEBev6HUwHs/hMDnv7C/sJiAa
ABoAABAAAAAAKABAAACwAAERcHFzcXNyc3Jwcn9fVM9PVL9PRL9fQDtfX0TPX1TPT0TPT0AAAABAAAAA
AD8APwAAUACwARABcAACEzNTM1IQUzFTMRISUhNSM1IwUjFSERIwJ2fvz+hv2K/H7+hgJ2AXr8fv6G/AF6
fvx+fvwBevx+/Px+AXoAAAAAgAAAAAEAAQAAAMAFgAAAREhEScGFREUFhchPgE1ETQmIyEnIQYDgP0AY
h48LQMuLTw8Lf5pa/7ULQMA/gACAN8eLf1YLTwDAzwtAigvPYACAAAAAAASAN4AAQAAAAAAAABAAA
AAQAAAAAAAQAUAAEAAQAAAAAAAgAHABUAAQAAAAAAAwAUABwAAQAAAAAABAAUADAAAQAAAAA
ABQALAEQAAQAAAAAABgAUAE8AAQAAAAAACgAsAGMAAQAAAAAACwASAI8AAwABBAkAAAACAKEAA
wABBAkAAQAoAKMAAwABBAkAAgAOAMsAAwABBAkAAwAoANkAAwABBAkABAAoAQEAAwABBAkABQ
AWASkAAwABBAkABgAoAT8AAwABBAkACgBYAWcAAwABBAkACwAkAb8gY3VzdG9tLXRvb2xiYXJbMTkw
OF1SZWd1bGFyY3VzdG9tLXRvb2xiYXJbMTkwOF1jdXN0b20tdG9vbGJhclsxOTA4XVZlcnNpb24gMS4wY3V
zdG9tLXRvb2xiYXJbMTkwOF1Gb250IGdlbmVyYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3
dy5zeW5jZnVzaW9uLmNvbQAgAGMAdQBzAHQAbwBtAC0AdABvAG8AbABiAGEAcgBbADEAOQAwADgA
XQBSAGUAZwB1AGwAYQByAGMAdQBzAHQAbwBtAC0AdABvAG8AbABiAGEAcgBbADEAOQAwADgAXQB
```

jAHUAcwB0AG8AbQAtAHQAbwBvAGwAYgBhAHIAWwAxADkAMAA4AF0AVgBlAHIAcwBpAG8AbgAgADE
ALgAwAGMAdQBzAHQAbwBtAC0AdABvAG8AbABiAGEAcgBbADEAOQAwADgAXQBGAG8AbgB0ACAAZw
BlAG4AZQByAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQA
cgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAA
gAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAOAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAE
NAQ4BDwAIVG9wLWljb24LZG93bi1hcnJvdzIKUFZfWm9vbW91dAlQVl9ab29taW4LUFZfRG93bmxvYWQL
UFZfQm9va21hcmsJUFZfU2VhcmNoCFBWX1ByaW50C1BWX1ByZXZpb3VzB1BWX05leHQIUFZfQ2xvc2U
MUFZfRml0VG9QYWdlB1BWX09wZW4AAA==) format('truetype');

}

</style>

`

The icons are embedded in the font file used in above code snippet.

Step 8: Add the following scripts for performing user interaction in PDF Viewer in code behind

STANDALONE

```
import Vue from "vue";
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
"@syncfusion/ej2-vue-pdfviewer";
import { ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
import { CalendarPlugin } from '@syncfusion/ej2-vue-calendars';
Vue.use(CalendarPlugin);
Vue.use(PdfViewerPlugin);
Vue.use(ToolbarPlugin);
var toolbar;
var viewer;
var fileName;
var currentPageBox;
function updatePageNavigation() {
if (viewer.currentPageNumber === 1) {
toolbar.enableItems(document.getElementById('previousPage'), false);
toolbar.enableItems(document.getElementById('nextPage'), true);
}
else if (viewer.currentPageNumber === viewer.pageCount) {
toolbar.enableItems(document.getElementById('previousPage'), true);
toolbar.enableItems(document.getElementById('nextPage'), false);
}
else {
toolbar.enableItems(document.getElementById('previousPage'), true);
toolbar.enableItems(document.getElementById('nextPage'), true);
}
}
function onCurrentPageBoxKeypress(event) {
if ((event.which < 48 || event.which > 57) && event.which !== 8 &&
event.which !== 13) {
event.preventDefault();
return false;
}
else {
var currentPageNumber = parseInt(currentPageBox.value);
```

```
if (event.which === 13) {
if (currentPageNumber > 0 && currentPageNumber <= viewer.pageCount) {
viewer.navigation.goToPage(currentPageNumber);
}
else {
currentPageBox.value = viewer.currentPageNumber.toString();
}
}
return true;
}
}
function onCurrentPageBoxClicked() {
currentPageBox.select();
}
function readFile(args) {
// tslint:disable-next-line
let upoadedFiles = args.target.files;
if (args.target.files[0] !== null) {
let uploadedFile = upoadedFiles[0];
fileName = uploadedFile.name;
if (uploadedFile) {
let reader = new FileReader();
reader.readAsDataURL(uploadedFile);
// tslint:disable-next-line
reader.onload = function (e) {
let uploadedFileUrl = e.currentTarget.result;
viewer.load(uploadedFileUrl, null);
viewer.fileName = fileName;
};
}
}
};
export default Vue.extend({
data: function () {
return {
documentPath: "https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf",
enableToolbar: false,
pageNoTemplate: '<div class=""><input type="text" class="e-input-group e-pv-
current-page-number" id="currentPage" /></div>',
pageTextTemplate: '<div class=""><span class="e-pv-total-page-number"
id="totalPage">of 0</span></div>'
}
},
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
openClicked: function (args) {
document.getElementById('fileUpload').click();
},
previousClicked: function (args) {
viewer.navigation.goToPreviousPage();
},
nextClicked: function (args) {
viewer.navigation.goToNextPage();
},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
printClicked: function (args) {
viewer.print.print();
},
downloadClicked: function (args) {
viewer.download();
},
pageFitClicked: function (args) {
viewer.magnification.fitToPage();
},
zoomInClicked: function (args) {
viewer.magnification.zoomIn();
},
zoomOutClicked: function (args) {
viewer.magnification.zoomOut();
},
pageChange: function (args) {
currentPageBox = document.getElementById('currentPage');
currentPageBox.value = viewer.currentPageNumber.toString();
updatePageNavigation();
},
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
toolbar = this.$refs.toolbar.ej2Instances;
currentPageBox = document.getElementById('currentPage');
currentPageBox.value = '1';
document.getElementById('totalPage').textContent = 'of ' + viewer.pageCount;
currentPageBox.addEventListener('keypress', onCurrentPageBoxKeypress);
currentPageBox.addEventListener('click', onCurrentPageBoxClicked);
document.getElementById('fileUpload').addEventListener('change', readFile,
false);
updatePageNavigation();
}
}
});
```

SERVER-BACKED

```
import Vue from "vue";
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
"@syncfusion/ej2-vue-pdfviewer";
import { ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
import { CalendarPlugin } from '@syncfusion/ej2-vue-calendars';
Vue.use(CalendarPlugin);
Vue.use(PdfViewerPlugin);
Vue.use(ToolbarPlugin);
var toolbar;
var viewer;
var fileName;
var currentPageBox;
function updatePageNavigation() {
if (viewer.currentPageNumber === 1) {
toolbar.enableItems(document.getElementById('previousPage'), false);
toolbar.enableItems(document.getElementById('nextPage'), true);
}
```

```
else if (viewer.currentPageNumber === viewer.pageCount) {
toolbar.enableItems(document.getElementById('previousPage'), true);
toolbar.enableItems(document.getElementById('nextPage'), false);
}
else {
toolbar.enableItems(document.getElementById('previousPage'), true);
toolbar.enableItems(document.getElementById('nextPage'), true);
}
}
function onCurrentPageBoxKeypress(event) {
if ((event.which < 48 || event.which > 57) && event.which !== 8 &&
event.which !== 13) {
event.preventDefault();
return false;
}
else {
var currentPageNumber = parseInt(currentPageBox.value);
if (event.which === 13) {
if (currentPageNumber > 0 && currentPageNumber <= viewer.pageCount) {
viewer.navigation.goToPage(currentPageNumber);
}
else {
currentPageBox.value = viewer.currentPageNumber.toString();
}
}
return true;
}
}
function onCurrentPageBoxClicked() {
currentPageBox.select();
}
function readFile(args) {
// tslint:disable-next-line
let upoadedFiles = args.target.files;
if (args.target.files[0] !== null) {
let uploadedFile = upoadedFiles[0];
fileName = uploadedFile.name;
if (uploadedFile) {
let reader = new FileReader();
reader.readAsDataURL(uploadedFile);
// tslint:disable-next-line
reader.onload = function (e) {
let uploadedFileUrl = e.currentTarget.result;
viewer.load(uploadedFileUrl, null);
viewer.fileName = fileName;
};
}
}
};
export default Vue.extend({
data: function () {
return {
serviceUrl: "https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath: "https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf",
enableToolbar: false,
pageNoTemplate: '<div class=""><input type="text" class="e-input-group e-pv-
current-page-number" id="currentPage" /></div>',
```

```
pageTextTemplate: '<div class=""><span class="e-pv-total-page-number"
id="totalPage">of 0</span></div>'
}
},
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
openClicked: function (args) {
document.getElementById('fileUpload').click();
},
previousClicked: function (args) {
viewer.navigation.goToPreviousPage();
},
nextClicked: function (args) {
viewer.navigation.goToNextPage();
},
printClicked: function (args) {
viewer.print.print();
},
downloadClicked: function (args) {
viewer.download();
},
pageFitClicked: function (args) {
viewer.magnification.fitToPage();
},
zoomInClicked: function (args) {
viewer.magnification.zoomIn();
},
zoomOutClicked: function (args) {
viewer.magnification.zoomOut();
},
pageChange: function (args) {
currentPageBox = document.getElementById('currentPage');
currentPageBox.value = viewer.currentPageNumber.toString();
updatePageNavigation();
},
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
toolbar = this.$refs.toolbar.ej2Instances;
currentPageBox = document.getElementById('currentPage');
currentPageBox.value = '1';
document.getElementById('totalPage').textContent = 'of ' + viewer.pageCount;
currentPageBox.addEventListener('keypress', onCurrentPageBoxKeypress);
currentPageBox.addEventListener('click', onCurrentPageBoxClicked);
document.getElementById('fileUpload').addEventListener('change', readFile,
false);
updatePageNavigation();
}
}
});
```

Sample :

https://ej2.syncfusion.com/vue/demos/#/material/pdfviewer/custom-toolbar.html

Copyright © 2001 -2024 Syncfusion Inc.

Load documents in Vue Pdfviewer component

The PDF Viewer library allows you to switch or load PDF documents dynamically after the initial load operation. To achieve this, load the PDF document as a base64 string or file name into the PDF Viewer control using the **Load()** method dynamically.

The following steps are used to load the PDF document dynamically.

Step 1: Follow the steps provided in the link to create a simple PDF Viewer sample.

Step 2: Use the following code snippet to load the PDF document using a base64 string.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="load">LoadDocument</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Load the PDF document from a Base64 string
load: function (event) {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.load('data:application/pdf;base64',+AddBase64String, null);
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div>
<button v-on:click="load_1">LoadDocumentFromBase64</button>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<div>
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView,ThumbnailView, Print,TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch,Annotation, FormDesigner, FormFields ]},
methods: {
// Load the PDF document from a Base64 string
load_1: function (event) {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.load('data:application/pdfbase64,'+ AddBase64String, null);
}
}
}
</script>
```

Step 3: Use the following code snippet to load the PDF document using the document name.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="load_2">LoadDocument</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
load_2: function (event) {
// Load the PDF document using the file name
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.load('https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf',
null);
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="load_2">LoadDocument</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
```

```
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
load_2: function (event) {
// Load the PDF document using the file name
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.load('https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf',
null);
}
}
}
</script>
```

Find the sample, how to load PDF documents dynamically

Unload document in Vue Pdfviewer component

The PDF Viewer library allows you to unload the PDF document being displayed in the PDF Viewer control programmatically using the **unload()** method.

The following steps are used to unload the PDF document programmatically.

Step 1: Follow the steps provided in the link to create a simple PDF Viewer sample.

Step 2: Add the following code snippet to perform the unload operation.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="unload">Unload Document</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
```

```
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
unload: function() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
// Unload the PDF document.
viewer.unload();
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="unload">Unload Document</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
unload: function() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
// Unload the PDF document.
viewer.unload();
}
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

Find the Sample, how to unload the PDF document programmatically

Open thumbnail in Vue Pdfviewer component

The PDF Viewer library allows you to open the thumbnail of the PDF document using the **openThumbnailPanel()** method.

The following steps are used to open the thumbnail of the PDF Document.

Step 1: Follow the steps provided in the link to create simple PDF Viewer sample in Vue.

Step 2: Add the following code snippet to open the thumbnail in the resizing event of splitter.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="onSplitterResize">Open ThumbnailPanel</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath" >
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
onSplitterResize: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.updateViewerContainer();
viewer.thumbnailViewModule.openThumbnailPane();
debugger;
},
}
}
</script>
```

SERVER-BACKED

```
<template>
```

```
<div id="app">
<button v-on:click="onSplitterResize">Open ThumbnailPanel</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
onSplitterResize: function () {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.updateViewerContainer();
viewer.thumbnailViewModule.openThumbnailPane();
debugger;
},
}
}
</script>
```

Find the Sample how to open thumbnail

Extract text in Vue Pdfviewer component

The PDF Viewer library allows you to extract the text from a page along with the bounds. Text extraction can be done using the **isExtractText** property and **extractTextCompleted** event.

The following steps are used to extract the text from the page.

Step 1: Follow the steps provided in the link to create a simple PDF Viewer sample.

Step 2: The following code snippet explains how to extract the text from a page .

`

<template>

```
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:isExtractText="true"
:extractTextCompleted="extractTextCompleted">
</ejs-pdfviewer>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView,ThumbnailView, Print,TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"PDF_Succinctly.pdf",
extractTextCompleted: function (args) {
// Extract the Complete text of load document
console.log(args);
console.log(args.documentTextCollection[1]);
// Extract the Text data.
console.log(args.documentTextCollection[1][1].TextData);
// Extract Text in the Page.
console.log(args.documentTextCollection[1][1].PageText);
// Extracts the first text of the PDF document along with its bounds
console.log(args.documentTextCollection[1][1].TextData[0].Bounds);
}
};
},
provide: {
```

PdfViewer: [Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView,

Print, TextSelection, TextSearch,Annotation, FormDesigner, FormFields]}

}

</script>

`

Find the Sample, how to Extract Text

Import export annotation object in Vue Pdfviewer component

The PDF Viewer library allows you to import annotations from objects or streams instead of loading it as a file. To import such annotation objects, the PDF Viewer control must export the PDF annotations as objects using the **ExportAnnotationsAsObject()** method. Only the annotations objects that are exported from the PDF Viewer can be imported.

The following steps are used to import and export annotation as object.

Step 1: Follow the steps provided in the link to create a simple PDF Viewer sample.

Step 2: Use the following code snippet to perform import and export annotation.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="exportAnnotation">Export Annotation</button>
<button v-on:click="importAnnotation">Import Annotation</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var exportObject;
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
//Export annotation as object.
```

```
exportAnnotation: function() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.exportAnnotationsAsObject().then(function(value) {
exportObject = value;
});
},
//Import annotation that are exported as object.
importAnnotation: function() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.importAnnotation(JSON.parse(exportObject));
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="exportAnnotation">Export Annotation</button>
<button v-on:click="importAnnotation">Import Annotation</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
var exportObject;
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
//Export annotation as object.
exportAnnotation: function() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.exportAnnotationsAsObject().then(function(value) {
exportObject = value;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
});
},
//Import annotation that are exported as object.
importAnnotation: function() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.importAnnotation(JSON.parse(exportObject));
}
}
}
</script>
```

Find the Sample, how to import and export annotation as object

Authorization token in Vue Pdfviewer component

The PDF Viewer library allows you to include the authorization token in the PDF viewer AJAX request using the properties of the ajaxRequest header available in AjaxRequestSettings, and it will be included in every AJAX request send from PDF Viewer.

The following steps are used to include the authorization token to the PDF viewer control.

Step 1: Follow the steps provided in the link to create simple PDF Viewer sample in React.

Step 2: Add the following code snippet to include the authorization token.

`

<template>

<ejs-pdfviewer

id="pdfViewer"

:serviceUrl="serviceUrl"

:documentPath="documentPath"

:ajaxRequestSettings="ajaxRequestSettings">

</ejs-pdfviewer>

</template>

<script>

import Vue from 'vue';

import { PdfViewerPlugin, Toolbar, Magnification, Navigation, LinkAnnotation,

BookmarkView,ThumbnailView, Print,TextSelection, TextSearch,

Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';

Vue.use(PdfViewerPlugin);

export default {

data () {

```
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"PDF_Succinctly.pdf",
// Include Authorization Token.
ajaxRequestSettings:{
ajaxHeaders: [
{
headerName: 'Authorization',
headerValue: 'Bearer 64565dfgfdsjweiuvbiuyhiueygf'
}
],
withCredentials: false
},
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]}
}
</script>
`
```

Find the Sample, how to include authorization token

Delete annotation in Vue Pdfviewer component

The PDF Viewer library allows you to delete a specific annotation from a PDF document. Deleting a specific annotation can be done using the **deleteAnnotationById()** method. This method is used to delete a specific annotation using its id.

The following steps are used to delete a specific annotation from PDF Document.

Step 1: Follow the steps provided in the link to create a simple PDF Viewer sample.

Step 2: Use the following code snippet to delete a specific annotation using deleteAnnotationById() method.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="deleteAnnotationById">Delete Annotation By Id</button>
<ejs-pdfviewer
id="pdfViewer"
```

```
ref="pdfviewer"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Delete annotation by ID.
deleteAnnotationById: function() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.annotationModule.deleteAnnotationById(viewer.annotationCollection[0].
annotationId);
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="deleteAnnotationById">Delete Annotation By Id</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
// Delete annotation by ID.
deleteAnnotationById: function() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.annotationModule.deleteAnnotationById(viewer.annotationCollection[0].
annotationId);
}
}
}
</script>
```

Note :

If you use the **deleteAnnotation** API to delete an annotation, select an annotation first using the **selectAnnotation** API. You can't delete an annotation without selecting that annotation.

Find the sample, how to delete a specific annotation using deleteAnnotationById

View sample in GitHub

Convert pixel to point in server side in Vue Pdfviewer component

Syncfusion PDF viewer will get the bounds of the annotations as the pixel in the Client-side. But while using it in the back end, we need to convert the pixel into point by using the below calculation. And the 0.75 is constant for all the calculations in the back end.

`ts

X = x * 72 / 96

Y = y * 72 / 96

Width = width * 72 / 96

Height = height * 72 / 96

`

Select annotation in Vue Pdfviewer component

The PDF Viewer library allows you to select an annotation from a PDF document. Selecting an annotation can be done using the selectAnnotation() method. This method is used to select an annotation using its id.

The following steps are used to select an annotation from a PDF Document.

Step 1: Follow the steps provided in this link to create a simple PDF Viewer sample.

Copyright © 2001 -2024 Syncfusion Inc.

Step 2: The following code sample represents how to select an annotation in a button click.

STANDALONE

```
<template>
<div id="app">
<button v-on:click="selectAnnotation">SelectAnnotation</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
selectAnnotation: function() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
// Unload the PDF document.
viewer.annotationModule.selectAnnotation(viewer.annotationCollection[0].anno
tationId);
}
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<button v-on:click="selectAnnotation">SelectAnnotation</button>
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:serviceUrl="serviceUrl"
:documentLoad="documentLoad">
</ejs-pdfviewer>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
serviceUrl:"https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf"
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
methods: {
selectAnnotation: function() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
// Unload the PDF document.
viewer.annotationModule.selectAnnotation(viewer.annotationCollection[0].anno
tationId);
}
}
}
</script>
```

`

```
<button id="selectAnnotation">SelectAnnotation</button>

<script>

//Event triggers when you click the SelectAnnotation button.

document.getElementById('selectAnnotation').addEventListener('click', () => {

//API to select the annotation

viewer.annotationModule.selectAnnotation(viewer.annotationCollection[0].annotationId);

});

</script>
```

`

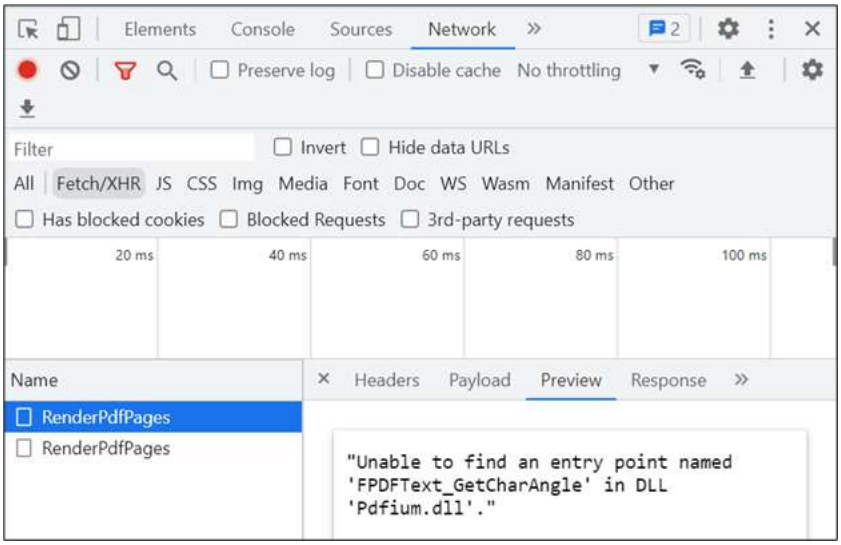
[View sample in GitHub]

Resolve "Unable to find an entry point named FPDFText_GetCharAngle" error

From the release of version **21.1.0.35 (2023 Volume 1)** of Essential Studio, the Pdfium package has been upgraded to improve various functionalities like text search, text selection, rendering, and even

performance. If you are updating your project to this version of the Syncfusion PDF Viewer, you may encounter the **“Web-Service is not listening”** error. The Network tab can help you identify the root cause of the issue, which is typically caused by an old version of pdfium assembly being referenced in the local web service project. Below are the assemblies to be referred to in the respective operating systems.

- Windows – pdfium.dll
- Linux – libpdfium.so
- OSX – libpdfium.dylib

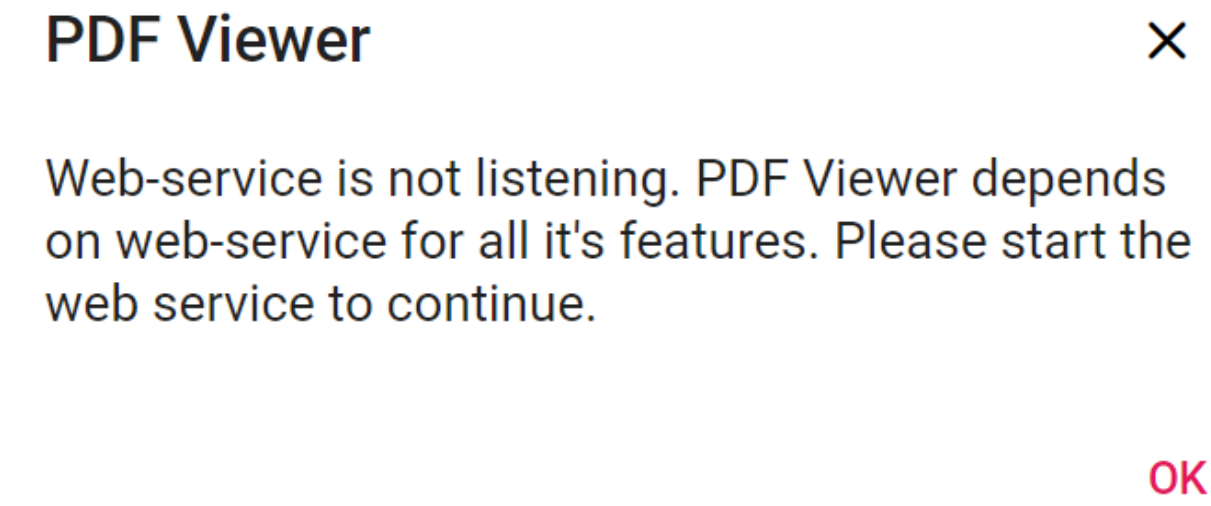


To solve this issue, you should follow the below steps:

1. Clear the bin and object files of the web-service project.
2. Re-publish the web-service project.

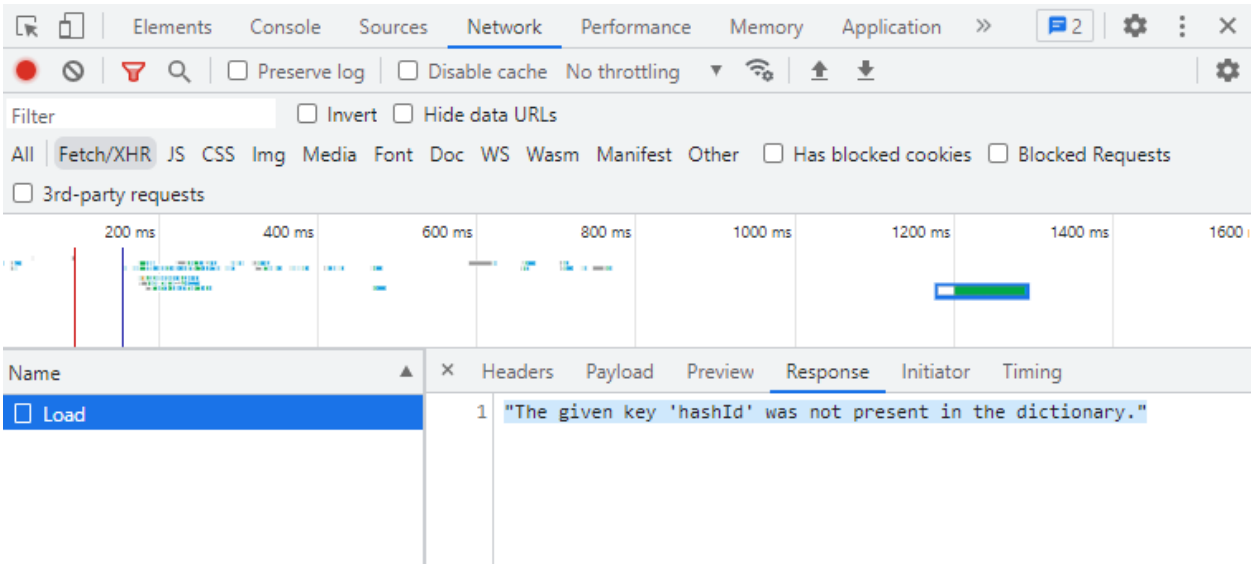
Note: Note: If you are hosting your application in Azure, AWS, or in Linux environments, delete the older published files and republish the application.

How to clear the "Web-service is not listening" to error



If you are facing a **Web-service is not listening** to error in the Syncfusion PDF Viewer, there could be several reasons for this. To troubleshoot the issue, you can use the Network tab in your browser's developer tools to gather more information. Here are the steps you can follow:

Step 1: Open the browser's developer tools by right-clicking on the page and selecting `Inspect` from the dropdown menu. Then Navigate to the `Network` tab. This will show you all of the requests that are being made by the page.



Step 2: Try to request the web service. If the service is not listening, the request will fail, and you should see an error message in the Network tab. Click on the failing request to see the details of the error, such as the error message or stack trace. This can help you identify the root cause of the issue. Check the server logs for any errors or warnings that may indicate the cause of the issue and help you to troubleshoot the problem.

Step 3: Check the request URL and parameters to see if they are correct. If there is a type or an incorrect parameter, the web service may be unable to process the request.

Copyright © 2001 -2024 Syncfusion Inc.

By following these steps and using the Network tab in your browser's developer tools, you can gather more information about the issue and troubleshoot the problem more effectively.

Note: Make sure you are connected to the internet and that your connection is stable. You can try accessing other websites or services to see if they are working, and make sure the URL you are using to access the web service is correct and properly formatted.

Here are some common exceptions

- File not found.
- Document cache not found.
- Document pointer does not exist in the cache.

File not found

If you are encountering an error message stating that the web service is not listening due to a file not being found in the Syncfusion PDF viewer, you can try the following steps to resolve the issue:

Check the file path

Ensure that the file path you use to access the PDF file is correct and that the file exists in that location. You will need to update the file path if the file does not exist.

Document cache not found

The Document cache not found exception in Syncfusion PDF Viewer typically occurs when the cache used to store the rendered pages of a PDF document is not found or has been deleted. This can happen if the cache directory is changed or deleted or if the application is running in a different environment than it was previously.

Check for multiple instances

It's possible that you have multiple instances of the Syncfusion PDF Viewer running simultaneously, which can cause issues with the document cache. To check for this, open the Task Manager on your computer and look for any instances of the Syncfusion PDF Viewer running. If you find multiple instances, try closing them all and reopening the viewer.

We can use Redis cache and distributive cache for this issue.

Check your network connection

Ensure that your network connection is stable and strong enough to support the web service you are trying to use. Sometimes, simply restarting the web service can resolve the issue. Try stopping and starting the service again to see if it resolves the problem.

The document pointer does not exist in the cache.

The Document pointer does not exist in the cache exception in the Syncfusion PDF Viewer usually occurs when there is an issue with loading or caching the PDF document. This error can be caused by a variety of reasons, including:

To clear this error in the Syncfusion PDF Viewer, you can try the following steps:

Step 1: Clearing the cache may help resolve the issue. To clear the cache, navigate to the cache location, which can be found in the Syncfusion PDF Viewer's settings or configuration files. Once you locate the cache folder, delete its contents.

Step 2: Try reloading the document to ensure it is loaded correctly. You can do this by calling the controller's Load() method. Ensure the document is not already loaded before attempting to load it again.

Copyright © 2001 -2024 Syncfusion Inc.

Step 3: Restart the application. If clearing the cache does not work, you can try restarting the PDF Viewer application. This will reload all the necessary components and may resolve the error.

Internal server error

Server-side exceptions happen for various use cases. We can't just define them if they are document-specific, provide the document, or you may need to contact Syncfusion support for further assistance.

Retry Timeout

The **Retry Timeout** feature allows you to specify a duration for the PDF Viewer to retry failed AJAX requests before considering them permanent failures. It helps in handling temporary failures due to network issues or server unavailability.

The retryTimeout allows developers to specify a duration after which the AJAX request should retry failed requests automatically. Developers can ensure a more resilient and fault-tolerant PDF viewing experience by configuring an appropriate retry timeout value.

By default, when an AJAX request fails, the Retry Timeout property is set to `0`, indicating that no timeout is set. In this case, the PDF Viewer will wait indefinitely for a response, potentially leading to a hanging request. However, you can set the Retry Timeout property to a positive number, specifying the maximum number of seconds the PDF Viewer should wait for a response. If the response is not received within the specified time, the request will be aborted, triggering an appropriate error or timeout property.

To set the retry timeout, use the `retryTimeout` property in the PDF Viewer configuration. This property takes a value in seconds.

`

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
:serviceUrl="serviceUrl"
:documentPath="documentPath"
:retryTimeout="retryTimeout"
:retryCount="retryCount" >
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
Annotation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from '@syncfusion/ej2-vue-pdfviewer';;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data() {
return {
serviceUrl: "https://services.syncfusion.com/vue/production/api/pdfviewer",
documentPath: "PDF_Succinctly.pdf",
retryTimeout : 10,
retryCount : 10
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]},
}
</script>
`
```

In the given example, the `retryTimeout` is set to 10 seconds, and the `retryCount` is set to 5. This means that if a request made by the PDF Viewer takes longer than 10 seconds to receive a response, it will be considered a timeout. In such cases, The PDF Viewer will resend the same request based on the retryCount. Here, this process will repeat up to maximum of 5 retries.

When an exception occurs during the AJAX request in the context of the PDF Viewer, the request will wait for the specified `retryTimeout` duration. If the timeout duration is exceeded, the PDF Viewer will decrement the retryCount and attempt to load the document again. This retry process continues until the document is successfully loaded or the retryCount limit is reached.

The `retryCount` property of the PDF Viewer allows you to set the number of retries for a specific request. This feature is particularly useful for handling temporary errors such as network timeouts or server issues. By initiating new requests according to the retry count, ensure a smoother user experience and efficiently handle network or server problems.

If the timeout duration specified by `retryTimeout` is exceeded during the AJAX request, the PDF Viewer will decrement the `retryCount` and initiate a new request. This process will continue until the document is successfully loaded or the retry count limit is reached. This functionality helps improve the handling of temporary errors and ensures a more efficient and user-friendly experience.

View sample in GitHub

Load N number of pages on initial loading

The initial rendering in a PDF viewer allows users to control the number of pages displayed when opening a PDF document. This improves the user experience by providing flexibility in loading a specific

number of pages initially, while additional pages are dynamically rendered as the user scrolls through the document. This approach enhances the responsiveness of the PDF viewer and minimizes delays as users can access specific pages without waiting for the entire document to load.

To utilize this capability in Syncfusion PDF Viewer, use the initialRenderPages property. You can achieve the desired outcome by setting this property to the desired number of pages you want to load initially. However, it's important to exercise caution when setting a high value for the initialRenderPages in large documents with numerous pages. Rendering a large number of pages simultaneously can increase memory usage and slow down loading times, impacting the performance of the PDF viewer.

Using the initialRenderPages property judiciously is advisable, especially when dealing with larger documents. It is more suitable for scenarios where a smaller range of pages, such as 10-20, can be loaded to provide a quick initial view of the document.

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:initialRenderPages="initialRenderPages"  >
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
initialRenderPages : 10
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
```

Copyright © 2001 -2024 Syncfusion Inc.

```
:documentPath="documentPath"
:serviceUrl="serviceUrl"
:initialRenderPages="initialRenderPages"  >
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation, BookmarkView, Annotation, ThumbnailView,
Print, TextSelection, TextSearch, FormFields, FormDesigner } from
'@syncfusion/ej2-vue-pdfviewer';
Vue.use(PdfViewerPlugin);
export default {
name: 'app',
data () {
return {
documentPath:"https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: "https://services.syncfusion.com/vue/production/api/pdfviewer",
initialRenderPages : 10
};
},
provide: {
PdfViewer: [ Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, Annotation,
ThumbnailView, Print, TextSelection, TextSearch, FormFields, FormDesigner
]},
}
</script>
```

`

<template>

<ejs-pdfviewer

id="pdfViewer"

:serviceUrl="serviceUrl"

:documentPath="documentPath"

:initialRenderPages="initialRenderPages" >

</ejs-pdfviewer>

</template>

<script>

import Vue from 'vue';

import { PdfViewerPlugin, Toolbar, Magnification, Navigation,

Annotation, LinkAnnotation, BookmarkView, ThumbnailView,

Copyright © 2001 -2024 Syncfusion Inc.

Print, TextSelection, TextSearch, FormFields, FormDesigner } from '@syncfusion/ej2-vue-pdfviewer';

Vue.use(PdfViewerPlugin);

export default {

name: 'app',

data() {

return {

serviceUrl: "https://services.syncfusion.com/vue/production/api/pdfviewer",

documentPath: "PDF_Succinctly.pdf",

initialRenderPages : 10

};

},

provide: {

PdfViewer: [Toolbar, Magnification, Navigation, LinkAnnotation, BookmarkView, ThumbnailView,

Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields]},

}

</script>

`

View sample in GitHub

Customize toolbar in PDF Viewer component

How to customize existing toolbar in PDF Viewer component

PDF Viewer allows you to customize(add, show, hide, enable, and disable) existing items in a toolbar.

- Add - New items can defined by **CustomToolbarItemModel** and with existing items in **ToolbarSettings** property. Newly added item click action can be defined in toolbarclick.
- Show, Hide - Existing items can be shown or hidden using the ToolbarSettings property. Pre-defined toolbar items are available with ToolbarItem.
- Enable, Disable - Toolbar items can be enabled or disable using enabletoolbaritem

STANDALONE

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad"
:resourceUrl="resourceUrl"
:toolbarClick="toolbarClick"
:OnCreateSearch="OnCreateSearch"
:toolbarSettings="toolbarSettings">
</ejs-pdfviewer>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
import { ComboBox } from "@syncfusion/ej2-dropdowns";
import { TextBox } from "@syncfusion/ej2-inputs";
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
// Move the toolItem declaration inside the data function
var toolItem1 = {
prefixIcon: 'e-icons e-paste',
id: 'print',
tooltipText: 'Custom toolbar item',
align: 'left'
};
var toolItem2 = {
id: 'download',
text: 'Save',
tooltipText: 'Custom toolbar item',
align: 'right'
};
var LanguageList = ['Typescript', 'Javascript', 'Angular', 'C#', 'C',
'Python'];
var toolItem3 = {
type: 'Input',
tooltipText: 'Language List',
cssClass: 'percentage',
align: 'Left',
id: 'dropdown',
template: new ComboBox({ width: 100, value: 'TypeScript', dataSource:
LanguageList, popupWidth: 85, showClearButton: false, readonly: false })
};
var toolItem4 = {
type: 'Input',
tooltipText: 'Text',
align: 'Right',
cssClass: 'find',
id: 'textbox',
template: new TextBox({ width: 125, placeholder: 'Type Here'})
}
return {
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
resourceUrl: "https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2-pdfviewer-
lib",
toolbarSettings: {
toolbarItems: [toolItem1, toolItem2, 'OpenOption', 'PageNavigationTool',
'MagnificationTool', toolItem3, 'PanTool', 'SelectionTool', 'SearchOption',
'PrintOption', 'DownloadOption', 'UndoRedoTool', 'AnnotationEditTool',
'FormDesignerEditTool', toolItem4, 'CommentTool', 'SubmitForm']
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
},
provide: {
PdfViewer: [Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]
},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
},
toolbarClick: function (args) {
if (args.item && args.item.id === 'print') {
viewer.printModule.print();
}
else if (args.item && args.item.id === 'download') {
viewer.download();
}
},
}
}
</script>
```

SERVER-BACKED

```
<template>
<div id="app">
<ejs-pdfviewer
id="pdfViewer"
ref="pdfviewer"
:documentPath="documentPath"
:documentLoad="documentLoad"
:serviceUrl="serviceUrl"
:toolbarClick="toolbarClick"
:OnCreateSearch="OnCreateSearch"
:toolbarSettings="toolbarSettings">
</ejs-pdfviewer>
</div>
</template>
<script>
import Vue from 'vue';
import { PdfViewerPlugin, Toolbar, Magnification, Navigation,
LinkAnnotation,
BookmarkView, ThumbnailView, Print, TextSelection, TextSearch,
Annotation, FormDesigner, FormFields } from '@syncfusion/ej2-vue-pdfviewer';
import { ComboBox } from "@syncfusion/ej2-dropdowns";
import { TextBox } from "@syncfusion/ej2-inputs";
Vue.use(PdfViewerPlugin);
var viewer;
export default {
name: 'app',
data () {
// Move the toolItem declaration inside the data function
var toolItem1 = {
prefixIcon: 'e-icons e-paste',
id: 'print',
```

```
tooltipText: 'Custom toolbar item',
align: 'left'
};
var toolItem2 = {
id: 'download',
text: 'Save',
tooltipText: 'Custom toolbar item',
align: 'right'
};
var LanguageList = ['Typescript', 'Javascript', 'Angular', 'C#', 'C',
'Python'];
var toolItem3 = {
type: 'Input',
tooltipText: 'Language List',
cssClass: 'percentage',
align: 'Left',
id: 'dropdown',
template: new ComboBox({ width: 100, value: 'TypeScript', dataSource:
LanguageList, popupWidth: 85, showClearButton: false, readonly: false })
};
var toolItem4 = {
type: 'Input',
tooltipText: 'Text',
align: 'Right',
cssClass: 'find',
id: 'textbox',
template: new TextBox({ width: 125, placeholder: 'Type Here'})
}
return {
documentPath: "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf",
serviceUrl: "https://services.syncfusion.com/vue/production/api/pdfviewer",
toolbarSettings: {
toolbarItems: [toolItem1, toolItem2, 'OpenOption', 'PageNavigationTool',
'MagnificationTool', toolItem3, 'PanTool', 'SelectionTool', 'SearchOption',
'PrintOption', 'DownloadOption', 'UndoRedoTool', 'AnnotationEditTool',
'FormDesignerEditTool', toolItem4, 'CommentTool', 'SubmitForm']
}
};
},
provide: {
PdfViewer: [Toolbar, Magnification, Navigation, LinkAnnotation,
BookmarkView, ThumbnailView,
Print, TextSelection, TextSearch, Annotation, FormDesigner, FormFields ]
},
methods: {
documentLoad: function (args) {
viewer = this.$refs.pdfviewer.ej2Instances;
},
toolbarClick: function (args) {
if (args.item && args.item.id === 'print') {
viewer.printModule.print();
}
else if (args.item && args.item.id === 'download') {
viewer.download();
}
},
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
```

Note : Default value of toolbar items is ['OpenOption', 'PageNavigationTool','MagnificationTool', 'PanTool', 'SelectionTool', 'SearchOption', 'PrintOption', 'DownloadOption','UndoRedoTool', 'AnnotationEditTool', 'FormDesignerEditTool', 'CommentTool', 'SubmitForm']

Align Property

The align property is used to specify the alignment of a toolbar item within the toolbar.

Left: Aligns the item to the left side of the toolbar.

Right: Aligns the item to the right side of the toolbar.

Tooltip Property

The tooltip property is used to set the tooltip text for a toolbar item. Tooltip provides additional information when a user hovers over the item.

CssClass Property

The cssClass property is used to apply custom CSS classes to a toolbar item. It allows custom styling of the toolbar item.

Prefix Property

The prefix property is used to set the CSS class or icon that should be added as a prefix to the existing content of the toolbar item.

ID Property

The id property within a CustomToolbarItemModel is a compulsory attribute that plays a vital role in toolbar customization. It serves as a unique identifier for each toolbar item, facilitating distinct references and interactions.

When defining or customizing toolbar items, it is mandatory to assign a specific and descriptive id to each item.

These properties are commonly used when defining custom toolbar items with the CustomToolbarItemModel in the context of Syncfusion PDF Viewer. When configuring the toolbar using the ToolbarSettings property, you can include these properties to customize the appearance and behavior of each toolbar item.

Note: When customizing toolbar items, you have the flexibility to include either icons or text based on your design preference.

View sample in GitHub

Troubleshooting

Why Do I Have to Manually Copy Files from node_modules into My App?

PDF Viewer offers flexibility across different build systems, remaining both framework-agnostic and independent of bundlers. Even without a bundler, you can seamlessly integrate the PDF Viewer by directly linking its assets through standard HTML tags.

Moreover, our codebase is meticulously divided into distinct files, enabling selective loading of components when required. This strategic approach to lazy loading prevents unwieldy file sizes that a single bundle might impose, which is often impractical.

Copyright © 2001 -2024 Syncfusion Inc.

While 'pdfium.js,' the primary entry point, is commonly bundled automatically, the supplementary assets from 'ej2-pdfviewer-lib' need to be manually incorporated due to their on-demand loading. This necessity arises because the host application lacks inherent awareness of these assets' lazy loading behavior.

Troubleshoot error 'cp' is not recognized as a command

The error message you're seeing, "'cp' is not recognized as an internal or external command," is because the `cp` command you're trying to use is not recognized on Windows command prompt.

On Windows, you should use the `copy` command to copy files and directories instead of `cp`. The equivalent command in Windows to copy a directory and its contents recursively is:

```batch
xcopy /s /e /i .\node_modules\@syncfusion\ej2-pdfviewer\dist\ej2-pdfviewer-lib public\js\ej2-pdfviewer-lib
```

Here, `/s` indicates that you want to copy directories and subdirectories recursively. Also, note that Windows uses backslashes `\` as path separators, not forward slashes `/`.

Make sure to run this command in the appropriate directory where you want to perform the copy operation.

Note: If you encounter other issues or error messages while working with the Windows Command Prompt, make sure to double-check your command syntax and file paths for accuracy. Additionally, ensure that you have the necessary permissions to perform the copy operation in the specified directories.

Document Loading Issues in Version 23.1 or Newer

If you're experiencing problems with your document not rendering in the viewer, especially when using version 23.1 or a newer version, follow these troubleshooting steps to resolve the issue:

1. **Check for viewer.dataBind() Requirement**: Ensure that you have called `viewer.dataBind()` as required in version 23.1 or newer. This explicit call is essential for initializing data binding and document rendering correctly. It is must to call the dataBind() method before load.

```typescript
<button v-on:click="documentLoad">Load</button>
<ejs-pdfviewer id="pdfViewer" ref="pdfviewer">
</ejs-pdfviewer>
documentLoad: function (args) {
pdfViewer.serviceUrl = "https://ej2services.syncfusion.com/angular/development/api/pdfviewer";
pdfViewer.dataBind();
pdfViewer.load("https://cdn.syncfusion.com/content/pdf/annotations.pdf");
}
```

Copyright © 2001 -2024 Syncfusion Inc.

2. **Verify Document Source**: Confirm that the document source or URL you're trying to display is valid and accessible. Incorrect URLs or document paths can lead to loading issues.
3. **Network Connectivity**: Ensure that your application has a stable network connection. Document rendering may fail if the viewer can't fetch the document due to network issues.
4. **Console Errors**: Use your browser's developer tools to check for any error messages or warnings in the console. These messages can provide insights into what's causing the document not to load.
5. **Loading Sequence**: Make sure that you're calling `viewer.dataBind()` and initiating document loading in the correct sequence. The viewer should be properly initialized before attempting to load a document.
6. **Update Viewer**: Ensure that you're using the latest version of the viewer library or framework. Sometimes, issues related to document loading are resolved in newer releases.
7. **Cross-Origin Resource Sharing (CORS)**: If you're loading documents from a different domain, ensure that CORS headers are correctly configured to allow cross-origin requests.
8. **Content Security Policies (CSP)**: Check if your application's Content Security Policy allows the loading of external resources, as this can affect document loading. Refer here to troubleshoot.

By following these troubleshooting steps, you should be able to address issues related to document loading in version 23.1 or newer, ensuring that your documents render correctly in the viewer.

Pivot Table

Getting Started with the Vue Pivotview Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Pivotview component

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies are required to use the pivot table component in your application.

`javascript

|-- @syncfusion/ej2-vue-pivotview

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-data

|-- @syncfusion/ej2-excel-export

|-- @syncfusion/ej2-file-utils

|-- @syncfusion/ej2-compression

|-- @syncfusion/ej2-pdf-export

|-- @syncfusion/ej2-file-utils

|-- @syncfusion/ej2-compression

|-- @syncfusion/ej2-grids

|-- @syncfusion/ej2-inputs

Copyright © 2001 -2024 Syncfusion Inc.

|-- @syncfusion/ej2-buttons

|-- @syncfusion/ej2-dropdowns

|-- @syncfusion/ej2-lists

|-- @syncfusion/ej2-popups

|-- @syncfusion/ej2-navigations

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

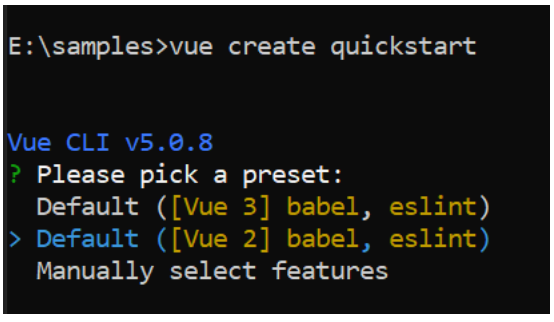
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Pivotview component as an example. Install the @syncfusion/ej2-vue-pivotview package by running the following command:

`bash

npm install @syncfusion/ej2-vue-pivotview --save

`

or

`bash

yarn add @syncfusion/ej2-vue-pivotview

`

The **--save** will instruct NPM to include the pivot table package inside the dependencies section of the package.json.

Import Syncfusion CSS styles

The pivot table has different themes. They are:

- Material
- Fabric
- Bootstrap
- High Contrast

import pivot table component CSS as given below in <style> section of the App.vue file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Pivotview component:

1\. First, import and register the Pivotview component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { PivotViewComponent } from "@syncfusion/ej2-vue-pivotview";
export default {
components: {
'ejs-pivotview': PivotViewComponent
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
}
</script>
```

2\. In the template section, define the Pivotview component.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-pivotview> </ejs-pivotview>
</div>
</template>
```

Browser compatibility

Polyfills are required to use the Pivot Table in Internet Explorer 11 browser. Refer the documentation for more details.

Assigning sample data to the pivot table

The Pivot Table component further needs to be populated with an appropriate data source. For illustration purpose, a collection of objects mentioning the sales details of certain products over a period and region has been prepared. This sample data is assigned to the pivot table component through dataSource property under dataSourceSettings.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-pivotview :dataSourceSettings="dataSourceSettings"> </ejs-pivotview>
</div>
</template>
<script>
import { PivotViewComponent } from "@syncfusion/ej2-vue-pivotview";
export default {
components: {
'ejs-pivotview': PivotViewComponent
},
data () {
return {
dataSourceSettings: {
dataSource: [
{
"balance": 2430.87,
"quantity": 11,
"name": "Skinner Ward",
"gender": "male",
"company": "GROK",
"state": "New Jercy"
},
{
"balance": 3192.7,
"quantity": 15,
"name": "Gwen Dixon",
"gender": "female",
"company": "ICOLOGY",
```

Copyright © 2001 -2024 Syncfusion Inc.

```
"state": "Vetaikan"
},
{
"balance": 1663.84,
"quantity": 14,
"name": "Deena Gillespie",
"gender": "female",
"company": "OVERPLEX",
"state": "New Jercy"
}
]
}
}
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

Adding fields to row, column, value and filter axes

Now that pivot table is initialized and assigned with sample data, will further move to showcase the component by organizing appropriate fields in row, column, value and filter axes.

In dataSourceSettings, four major axes - rows, columns, values and filters plays a vital role in defining and organizing fields from the bound data source, to render the entire pivot table component in a desired format.

rows – Collection of fields that needs to be displayed in row axis of the pivot table.

columns – Collection of fields that needs to be displayed in column axis of the pivot table.

values – Collection of fields that needs to be displayed as aggregated numeric values in the pivot table.

filters - Collection of fields that would act as master filter over the data bound in row, column and value axes of the pivot table.

In-order to define each field in the respective axis, the following basic properties should be set.

- name: It allows to set the field name from the bound data source. It's casing should match exactly like in the data source and if not set properly, the pivot table will not be rendered.
- caption: It allows to set the field caption, which is the alias name of the field that needs to be displayed in the pivot table.
- type: It allows to set the summary type of the field. By default, **Sum** is applied.

In this illustration, "Year" and "Quarter" are added in column, "Country" and "Products" in row, and "Sold" and "Amount" in value section respectively.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" > </ejs-pivotview>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import { PivotViewComponent } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
export default {
  components: {
    'ejs-pivotview': PivotViewComponent
  },
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs88" %}

Applying formatting to a value field

Formatting defines a way in which values should be displayed. For example, format **“C”** denotes the values should be displayed in currency pattern. To do so, define the formatSetting with its name and format properties and add it to formatSettings. In this illustration, the name property is set as **Amount**, a field from value section and its format is set as currency. Likewise, we can set format for other value fields as well and add it to formatSettings.

Only fields from value section, which is in the form of numeric data values are applicable for formatting.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" > </ejs-pivotview>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import { PivotViewComponent } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
export default {
  components: {
    'ejs-pivotview': PivotViewComponent
  },
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs89" %}

Module injection

To create pivot table with additional features, inject the required modules. The modules that are available with basic functionality are as follows.

- `GroupingBar` - Inject this module to access grouping bar feature.
- `FieldList` - Inject this module to access pivot field list feature.
- `CalculatedField` - Inject this module to access calculated field feature.

Register the required array of modules under the key `pivotview` in the `provide` section within the `App.vue` file as shown below. On doing so, only the injected views will be loaded and displayed along with pivot table.

`

Copyright © 2001 -2024 Syncfusion Inc.

<script>

provide: {

pivotview: [GroupingBar, FieldList, CalculatedField]

}

</script>

`

Enable Pivot Field List

The field list allows to add or remove fields and also rearrange the fields between different axes, including column, row, value, and filter along with filter and sort options dynamically at runtime. It can be enabled by setting the showFieldList property to **true**. To know more about field list, refer here.

If the FieldList module is not injected, the Field List will not be rendered with the pivot table component.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import { PivotViewComponent, FieldList } from "@syncfusion/ej2-vue-
pivotview";
import { pivotData } from './pivotData.js';
export default {
  components: {
    'ejs-pivotview': PivotViewComponent
  },
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs90" %}

Enable Grouping Bar

The grouping bar feature automatically populates fields from the bound data source and allows end users to drag fields between different axes such as columns, rows, values, and filters, and alter pivot table at runtime. It also provides option to sort, filter and remove fields. It can be enabled by setting the showGroupingBar property to **true**. To know more about grouping bar, refer here.

If the GroupingBar module is not injected, the grouping bar will not be rendered with the pivot table component.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showGroupingBar="showGroupingBar"> </ejs-pivotview>
    </div>
</template>
<script>
import { PivotViewComponent, GroupingBar } from "@syncfusion/ej2-vue-
pivotview";
import { pivotData } from './pivotData.js';
export default {
  components: {
    'ejs-pivotview': PivotViewComponent
  },
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showGroupingBar: true
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
  },
  provide: {
        pivotview: [GroupingBar]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs91" %}

Exploring Filter Axis

The filter axis contains collection of fields that would act as master filter over the data bound in row, column and value axes of the pivot table. The fields along with filter members could be set to filter axis either through report via code behind or by dragging and dropping fields from other axes to filter axis via grouping bar or field list at runtime.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showGroupingBar="showGroupingBar"> </ejs-pivotview>
    </div>
</template>
<script>
import { PivotViewComponent, GroupingBar } from "@syncfusion/ej2-vue-
pivotview";
import { pivotData } from './pivotData.js';
export default {
  components: {
    'ejs-pivotview': PivotViewComponent
  },
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [{ name: 'Country' }]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      },
      height: 350,
      showGroupingBar: true
    }
  },
  provide: {
        pivotview: [GroupingBar]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs92" %}

Calculated field

The calculated field feature allows user to insert or add a new calculated field based on the available fields from the bound data source using basic arithmetic operators. The calculated field can be included in pivot table using the calculatedFieldSettings from code behind. Or else, calculated fields can be added at run time through the built-in dialog by just setting the allowCalculatedField property to **true** in pivot table. You will see a button enabled in the Field List UI automatically to invoke the calculated field dialog and perform necessary operation. To know more about calculated field, refer here.

If the CalculatedField module is not injected, the calculated field popup will not be rendered with the pivot table component. Moreover calculated field is applicable only for value fields.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="calculated-field-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Calculated Field</ejs-button>
        <ejs-pivotview id="pivotview" :height= "height"
:dataSourceSettings="dataSourceSettings"
:allowCalculatedField="allowCalculatedField"> </ejs-pivotview>
    </div>
</template>
<script>
import { PivotViewComponent, CalculatedField } from "@syncfusion/ej2-vue-
pivotview";
import { ButtonComponent } from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
export default {
  components: {
    'ejs-pivotview': PivotViewComponent,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    'ejs-button': ButtonComponent
  },
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }, { name: 'Total', caption: 'Total Amount', type:
'CalculatedField' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Sold)"' }]
      },
      height: 320,
      allowCalculatedField: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function() {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.calculatedFieldModule.createCalculatedFieldDialog();
    }
  },
  provide: {
        pivotview: [CalculatedField]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs93" %}

Run the application

The quickstart project is configured to compile and run the application in the browser. Use the following command to run the application.

Copyright © 2001 -2024 Syncfusion Inc.

`bash

npm run dev

`

Output will be displayed as follows.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import { PivotViewComponent, FieldList } from "@syncfusion/ej2-vue-
pivotview";
import { pivotData } from './pivotData.js';
export default {
  components: {
    'ejs-pivotview': PivotViewComponent
  },
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Sold)"' }]
      },
      showFieldList: true,
      height: 350
    }
  },
  provide: {
    pivotview: [FieldList]
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs94" %}

You can also explore our Vue Pivot Table example that shows how to rendering of the pivot table with drill-up and drill-down functionality bound to a relational report.

Getting Started with the Vue Pivot Table component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Pivotview component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1. Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2. Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3. Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4. Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Pivotview component as an example. To use the Vue Pivotview component in the project, the @syncfusion/ej2-vue-pivotview package needs to be installed using the following command:

Copyright © 2001 -2024 Syncfusion Inc.

`bash

npm install @syncfusion/ej2-vue-pivotview --save

`

or

`bash

yarn add @syncfusion/ej2-vue-pivotview

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Pivotview component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue Pivotview component using Composition API or Options API:

1. First, import and register the Pivotview component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { PivotViewComponent, FieldList, GroupingBar, CalculatedField } from
"@syncfusion/ej2-vue-pivotview";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
```

```
import { PivotViewComponent, FieldList, GroupingBar, CalculatedField } from
"@syncfusion/ej2-vue-pivotview";
//Component registeration.
export default {
name: "App",
components: {
"ejs-pivotview": PivotViewComponent
}
}
</script>
```

2. In the template section, define the Pivotview component with the dataSource property and column definitions.

~/SRC/APP.VUE

```
<template>
<ejs-pivotview :height="height" :width="width"
:dataSourceSettings="dataSourceSettings" :showFieldList="showFieldList"
:showGroupingBar="showGroupingBar"
:allowCalculatedField="allowCalculatedField"></ejs-pivotview>
</template>
```

3. Declare the values for the dataSource property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const dataSource = [
{ Amount: 5100, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 21, State: "Alberta" },
{ Amount: 1900, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 23, State: "Alberta" },
{ Amount: 1000, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 29, State: "Alberta" },
{ Amount: 2060, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 93, State: "British Columbia" },
{ Amount: 6200, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 36, State: "British Columbia" },
{ Amount: 2000, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 31, State: "British Columbia" },
{ Amount: 1300, Country: "Canada", Date: "FY 2005", Product: "Car",
Quantity: 45, State: "Brunswick" },
{ Amount: 3400, Country: "France", Date: "FY 2006", Product: "Car",
Quantity: 47, State: "Brunswick" },
{ Amount: 2300, Country: "Germany", Date: "FY 2007", Product: "Car",
Quantity: 43, State: "Brunswick" },
]
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
```

```
dataSourceSettings: {
dataSource: [
{ Amount: 5100, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 21, State: "Alberta" },
{ Amount: 1900, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 23, State: "Alberta" },
{ Amount: 1000, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 29, State: "Alberta" },
{ Amount: 2060, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 93, State: "British Columbia" },
{ Amount: 6200, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 36, State: "British Columbia" },
{ Amount: 2000, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 31, State: "British Columbia" },
{ Amount: 1300, Country: "Canada", Date: "FY 2005", Product: "Car",
Quantity: 45, State: "Brunswick" },
{ Amount: 3400, Country: "France", Date: "FY 2006", Product: "Car",
Quantity: 47, State: "Brunswick" },
{ Amount: 2300, Country: "Germany", Date: "FY 2007", Product: "Car",
Quantity: 43, State: "Brunswick" },
],
rows: [{ name: 'Country' }],
columns: [{ name: 'Date' }],
values: [{ name: 'Amount' }, { name: 'Total', type: 'CalculatedField' }],
formatSettings: [{ name: 'Amount', format: 'C1' }],
calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Quantity)"' }]
},
showFieldList: true,
showGroupingBar: true,
allowCalculatedField: true,
height: '350px',
width: '100%'
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-pivotview :height="height" :width="width"
:dataSourceSettings="dataSourceSettings" :showFieldList="showFieldList"
:showGroupingBar="showGroupingBar"
:allowCalculatedField="allowCalculatedField"></ejs-pivotview>
</template>
<script setup>
export default {
data() {
return {
dataSourceSettings: {
dataSource: [
{ Amount: 5100, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 21, State: "Alberta" },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ Amount: 1900, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 23, State: "Alberta" },
{ Amount: 1000, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 29, State: "Alberta" },
{ Amount: 2060, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 93, State: "British Columbia" },
{ Amount: 6200, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 36, State: "British Columbia" },
{ Amount: 2000, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 31, State: "British Columbia" },
{ Amount: 1300, Country: "Canada", Date: "FY 2005", Product: "Car",
Quantity: 45, State: "Brunswick" },
{ Amount: 3400, Country: "France", Date: "FY 2006", Product: "Car",
Quantity: 47, State: "Brunswick" },
{ Amount: 2300, Country: "Germany", Date: "FY 2007", Product: "Car",
Quantity: 43, State: "Brunswick" },
],
rows: [{ name: 'Country' }],
columns: [{ name: 'Date' }],
values: [{ name: 'Amount' }, { name: 'Total', type: 'CalculatedField' }],
formatSettings: [{ name: 'Amount', format: 'C1' }],
calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Quantity)"' }]
},
showFieldList: true,
showGroupingBar: true,
allowCalculatedField: true,
height: '350px',
width: '100%'
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-pivotview :height="height" :width="width"
:dataSourceSettings="dataSourceSettings" :showFieldList="showFieldList"
:showGroupingBar="showGroupingBar"
:allowCalculatedField="allowCalculatedField">
</ejs-pivotview>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { PivotViewComponent, FieldList, GroupingBar, CalculatedField } from
"@syncfusion/ej2-vue-pivotview";
export default {
name: "App",
// Declaring component and its directives.
components: {
"ejs-pivotview": PivotViewComponent
},
// Bound properties declaration.
data() {
return {
dataSourceSettings: {
dataSource: [
{ Amount: 5100, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 21, State: "Alberta" },
{ Amount: 1900, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 23, State: "Alberta" },
{ Amount: 1000, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 29, State: "Alberta" },
{ Amount: 2060, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 93, State: "British Columbia" },
{ Amount: 6200, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 36, State: "British Columbia" },
{ Amount: 2000, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 31, State: "British Columbia" },
{ Amount: 1300, Country: "Canada", Date: "FY 2005", Product: "Car",
Quantity: 45, State: "Brunswick" },
{ Amount: 3400, Country: "France", Date: "FY 2006", Product: "Car",
Quantity: 47, State: "Brunswick" },
{ Amount: 2300, Country: "Germany", Date: "FY 2007", Product: "Car",
Quantity: 43, State: "Brunswick" },
],
rows: [{ name: 'Country' }],
columns: [{ name: 'Date' }],
values: [{ name: 'Amount' }, { name: 'Total', type: 'CalculatedField' }],
formatSettings: [{ name: 'Amount', format: 'C1' }],
calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Quantity)"' }]
},
showFieldList: true,
showGroupingBar: true,
allowCalculatedField: true,
height: '350px',
width: '100%'
};
},
// Injecting the required modules for additional features.
pivotview: [GroupingBar, FieldList, CalculatedField]
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

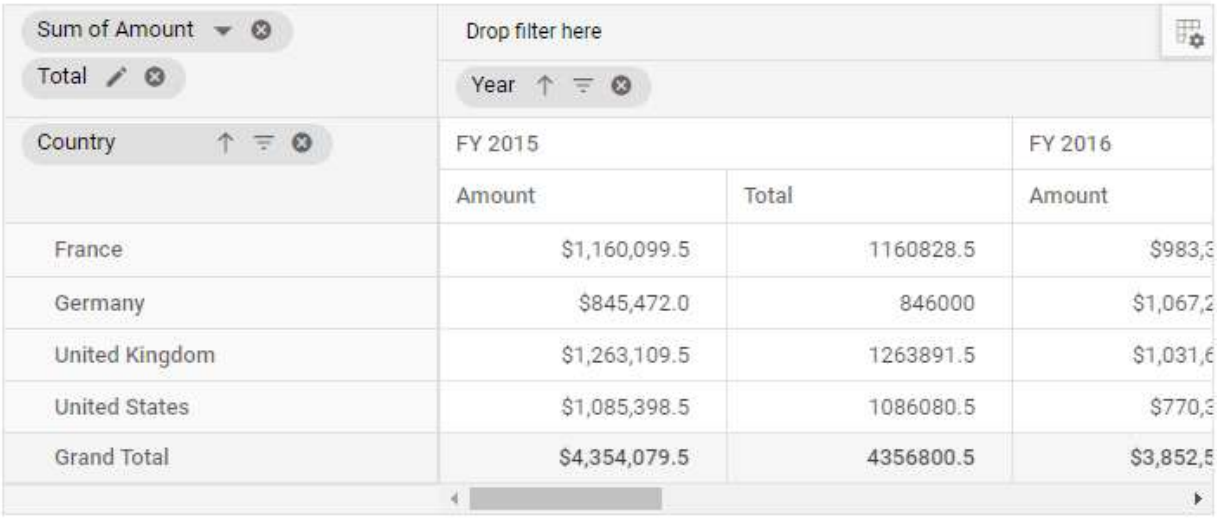
or

`bash

yarn run dev

`

The output will appear as follows:



Using template properties in the component

The following sample uses the HTML elements to showcase each cell with custom format using the Pivot Table's cellTemplate property.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height" :dataBound="trend"
:cellTemplate="cellTemplate">
</ejs-pivotview>
</template>
<script setup>
import { PivotViewComponent as EjsPivotview } from "@syncfusion/ej2-vue-
pivotview";
import { createApp } from 'vue';
```

```
const app = createApp();
var dataSourceSettings = {
dataSource: [
{ Amount: 5100, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 21, State: "Alberta" },
{ Amount: 1900, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 23, State: "Alberta" },
{ Amount: 1000, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 29, State: "Alberta" },
{ Amount: 2060, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 93, State: "British Columbia" },
{ Amount: 6200, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 36, State: "British Columbia" },
{ Amount: 2000, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 31, State: "British Columbia" },
{ Amount: 1300, Country: "Canada", Date: "FY 2005", Product: "Car",
Quantity: 45, State: "Brunswick" },
{ Amount: 3400, Country: "France", Date: "FY 2006", Product: "Car",
Quantity: 47, State: "Brunswick" },
{ Amount: 2300, Country: "Germany", Date: "FY 2007", Product: "Car",
Quantity: 43, State: "Brunswick" },
],
rows: [{ name: "Country" }],
columns: [{ name: "Date" }],
values: [{ name: "Amount" }, { name: "Total", type: 'CalculatedField' }],
formatSettings: [{ name: "Amount", format: "C1" }],
calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Quantity)"' }]
};
var colVue = app.component("cellTemplate", {
data() {
return {
data: {},
};
},
methods: {
getCellContent: function () {
return '<span class="tempwrap sb-icon-neutral pv-icons"></span>';
},
},
template: `<span class="template-wrap"><span class="tempwrap sb-icon-neutral
pv-icons"></span></span>`,
});
const cellTemplate = () => {
return { template: colVue };
};
const trend = () => {
let pivotGridObj = this.$refs.pivotview.ej2Instances;
var cTable = document.getElementsByClassName("e-table");
var colLen = pivotGridObj.pivotValues[3].length;
var cLen = cTable[3].children[0].children.length;
var rLen = cTable[3].children[1].children.length;
for (let k = 0; k < rLen; k++) {
if (
pivotGridObj.pivotValues[k] &&
pivotGridObj.pivotValues[k][0] !== undefined
) {
```

```
break;
}
}
var rowHeaders = [].slice.call(
cTable[2].children[1].querySelectorAll("td")
);
var rows = pivotGridObj.dataSourceSettings.rows;
if (rowHeaders.length > 1) {
for (var i = 0, Cnt = rows; i < Cnt.length; i++) {
var fields = {};
var fieldHeaders = [];
for (var j = 0, Lnt = rowHeaders; j < Lnt.length; j++) {
var header = rowHeaders[j];
if (
header.className.indexOf("e-gtot") === -1 &&
header.className.indexOf("e-rowsheader") > -1 &&
header.getAttribute("fieldname") === rows[i].name
) {
fields[rowHeaders[j].textContent] = j;
fieldHeaders.push(rowHeaders[j].textContent);
}
}
if (i === 0) {
for (var rnt = 0, Lnt1 = fieldHeaders; rnt < Lnt1.length; rnt++) {
if (rnt !== 0) {
var row = fields[fieldHeaders[rnt]];
var prevRow = fields[fieldHeaders[rnt - 1]];
for (var k = 0, ci = 1; k < cLen && ci < colLen; k++, ci++) {
var node = cTable[3].children[1].children[row].childNodes[k];
var prevNode =
cTable[3].children[1].children[prevRow].childNodes[k];
var ri = node.getAttribute("index");
var prevRi = prevNode.getAttribute("index");
if (ri < pivotGridObj.pivotValues.length) {
if (
pivotGridObj.pivotValues[prevRi][ci].value >
pivotGridObj.pivotValues[ri][ci].value &&
node.querySelector(".tempwrap")
) {
var trendElement = node.querySelector(".tempwrap");
trendElement.className = trendElement.className.replace(
"sb-icon-neutral",
"sb-icon-loss"
);
} else if (
pivotGridObj.pivotValues[prevRi][ci].value <
pivotGridObj.pivotValues[ri][ci].value &&
node.querySelector(".tempwrap")
) {
var trendElement1 = node.querySelector(".tempwrap");
trendElement1.className = trendElement1.className.replace(
"sb-icon-neutral",
"sb-icon-profit"
);
}
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
}
}
}
}
};
</script>
<style scoped>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-pivotview/styles/material.css";
#pivotview {
width: 100%;
}
@font-face {
font-family: "e-pivot";
src: /* To get the output shown below, add the resource containing font data
from the below reference sample. */
font-weight: normal;
font-style: normal;
}
.pv-icons {
font-family: "e-pivot";
font-style: normal;
font-variant: normal;
font-weight: normal;
text-transform: none;
line-height: 1;
}
.sb-icon-profit::before {
content: "\e234";
padding-left: 30px;
margin: auto !important;
color: #219122;
size: 20px;
}
.sb-icon-neutral::before {
content: "\e84d";
padding-left: 30px;
margin: auto !important;
color: #82b5e9;
}
.sb-icon-loss::before {
content: "\e239";
padding-left: 30px;
margin: auto !important;
color: #ff2222;
}
.sb-sample-content-area {
min-height: 255px !important;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.control-section {
min-height: 255px !important;
}
.e-columnsheader .tempwrap.sb-icon-neutral.pv-icons{
display: none !important;
}
.e-rowsheader .tempwrap.sb-icon-neutral.pv-icons {
display: none !important;
}
.e-pivotview .e-grid .e-rowcell {
font-size: 13px;
padding-left: 8px;
}
.e-cellvalue {
margin-top: 8px;
display: flex;
margin-left: 30px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height" :dataBound="trend"
:cellTemplate="cellTemplate"></ejs-pivotview>
</template>
<script>
import { PivotViewComponent } from "@syncfusion/ej2-vue-pivotview";
import { createApp } from 'vue/dist/vue.esm-bundler';
const app = createApp();
// Template declaration.
var colVue = app.component("cellTemplate", {
data() {
return {
data: {},
};
},
methods: {
getCellContent: function () {
return '<span class="tempwrap sb-icon-neutral pv-icons"></span>';
},
},
template: `<span class="template-wrap"><span class="tempwrap sb-icon-neutral
pv-icons"></span></span>`,
});
export default {
name: "App",
components: {
"ejs-pivotview": PivotViewComponent,
},
data() {
return {
dataSourceSettings: {
dataSource: [
```

```
{ Amount: 5100, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 21, State: "Alberta" },
{ Amount: 1900, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 23, State: "Alberta" },
{ Amount: 1000, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 29, State: "Alberta" },
{ Amount: 2060, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 93, State: "British Columbia" },
{ Amount: 6200, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 36, State: "British Columbia" },
{ Amount: 2000, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 31, State: "British Columbia" },
{ Amount: 1300, Country: "Canada", Date: "FY 2005", Product: "Car",
Quantity: 45, State: "Brunswick" },
{ Amount: 3400, Country: "France", Date: "FY 2006", Product: "Car",
Quantity: 47, State: "Brunswick" },
{ Amount: 2300, Country: "Germany", Date: "FY 2007", Product: "Car",
Quantity: 43, State: "Brunswick" },
],
rows: [{ name: "Country" }],
columns: [{ name: "Date" }],
values: [{ name: "Amount" }, { name: "Total", type: 'CalculatedField' }],
formatSettings: [{ name: "Amount", format: "C1" }],
calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Quantity)"' }]
},
height: 350,
cellTemplate: function () {
return { template: colVue };
},
};
},
methods: {
trend: function () {
let pivotGridObj = this.$refs.pivotview.ej2Instances;
var cTable = document.getElementsByClassName("e-table");
var colLen = pivotGridObj.pivotValues[3].length;
var cLen = cTable[3].children[0].children.length;
var rLen = cTable[3].children[1].children.length;
for (let k = 0; k < rLen; k++) {
if (
pivotGridObj.pivotValues[k] &&
pivotGridObj.pivotValues[k][0] !== undefined
) {
break;
}
}
var rowHeaders = [].slice.call(
cTable[2].children[1].querySelectorAll("td")
);
var rows = pivotGridObj.dataSourceSettings.rows;
if (rowHeaders.length > 1) {
for (var i = 0, Cnt = rows; i < Cnt.length; i++) {
var fields = {};
var fieldHeaders = [];
for (var j = 0, Lnt = rowHeaders; j < Lnt.length; j++) {
var header = rowHeaders[j];
```

```
if (
header.className.indexOf("e-gtot") === -1 &&
header.className.indexOf("e-rowsheader") > -1 &&
header.getAttribute("fieldname") === rows[i].name
) {
fields[rowHeaders[j].textContent] = j;
fieldHeaders.push(rowHeaders[j].textContent);
}
}
if (i === 0) {
for (var rnt = 0, Lnt1 = fieldHeaders; rnt < Lnt1.length; rnt++) {
if (rnt !== 0) {
var row = fields[fieldHeaders[rnt]];
var prevRow = fields[fieldHeaders[rnt - 1]];
for (var k = 0, ci = 1; k < cLen && ci < colLen; k++, ci++) {
var node = cTable[3].children[1].children[row].childNodes[k];
var prevNode =
cTable[3].children[1].children[prevRow].childNodes[k];
var ri = node.getAttribute("index");
var prevRi = prevNode.getAttribute("index");
if (ri < pivotGridObj.pivotValues.length) {
if (
pivotGridObj.pivotValues[prevRi][ci].value >
pivotGridObj.pivotValues[ri][ci].value &&
node.querySelector(".tempwrap")
) {
var trendElement = node.querySelector(".tempwrap");
trendElement.className = trendElement.className.replace(
"sb-icon-neutral",
"sb-icon-loss"
);
} else if (
pivotGridObj.pivotValues[prevRi][ci].value <
pivotGridObj.pivotValues[ri][ci].value &&
node.querySelector(".tempwrap")
) {
var trendElement1 = node.querySelector(".tempwrap");
trendElement1.className = trendElement1.className.replace(
"sb-icon-neutral",
"sb-icon-profit"
);
}
}
}
}
}
}
}
},
},
};
</script>
<style scoped>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
```

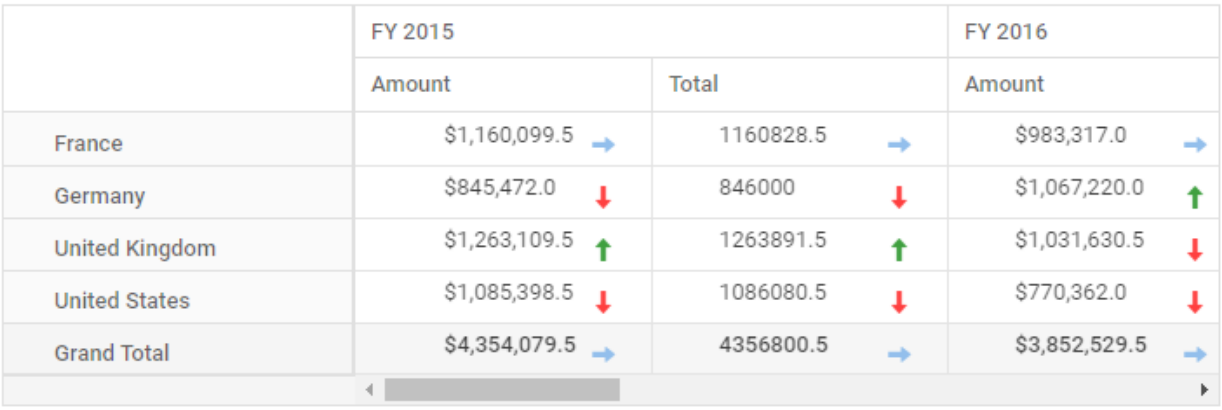
Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-pivotview/styles/material.css";
/deep/ #pivotview {
width: 100%;
}
@font-face {
font-family: "e-pivot";
src: /* To get the output shown below, add the resource containing font data
from the below reference sample. */
font-weight: normal;
font-style: normal;
}
/deep/ .pv-icons {
font-family: "e-pivot";
font-style: normal;
font-variant: normal;
font-weight: normal;
text-transform: none;
line-height: 1;
}
/deep/ .sb-icon-profit::before {
content: "\e234";
padding-left: 30px;
margin: auto !important;
color: #219122;
size: 20px;
}
/deep/ .sb-icon-neutral::before {
content: "\e84d";
padding-left: 30px;
margin: auto !important;
color: #82b5e9;
}
/deep/ .sb-icon-loss::before {
content: "\e239";
padding-left: 30px;
margin: auto !important;
color: #ff2222;
}
/deep/ .sb-sample-content-area {
min-height: 255px !important;
}
/deep/ .control-section {
min-height: 255px !important;
}
/deep/ .e-columnsheader .tempwrap.sb-icon-neutral.pv-icons{
display: none !important;
}
/deep/ .e-rowsheader .tempwrap.sb-icon-neutral.pv-icons {
display: none !important;
}
/deep/ .e-pivotview .e-grid .e-rowcell {
font-size: 13px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
padding-left: 8px;
}
/deep/ .e-cellvalue {
margin-top: 8px;
display: flex;
margin-left: 30px;
}
</style>
```

The output will appear as follows:



	FY 2015		FY 2016
	Amount	Total	Amount
France	$1,160,099.5	1160828.5	$983,317.0
Germany	$845,472.0	846000	$1,067,220.0
United Kingdom	$1,263,109.5	1263891.5	$1,031,630.5
United States	$1,085,398.5	1086080.5	$770,362.0
Grand Total	$4,354,079.5	4356800.5	$3,852,529.5

Refer the following sample, vue3-pivottable-celltemplate.

Sample: vue-3-pivottable-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Data binding in Vue Pivotview component

JSON

For JSON data binding, the type property under dataSourceSettings needs to be set as JSON. By default, the default value is assumed as JSON.

Binding JSON data via local

In-order to bind local JSON data to the pivot table user can assign the local variable holding the JSON data to the dataSource property under dataSourceSettings.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
```

```
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs19" %}

Using local variable, the JSON data can also be bound to the pivot table using DataManager option with the help of JsonAdaptor. Here the instance of DataManager holding JSON data is assigned to dataSource property under dataSourceSettings. The use of DataManager is optional here.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
import { DataManager, JsonAdaptor, Query, ReturnOption } from
'@syncfusion/ej2-data';
Vue.use(PivotViewPlugin);
let remoteData = new DataManager({
  json: pivotData,
  adaptor: new JsonAdaptor()
});
export default {
  data () {
    return {
```

```
      dataSourceSettings: {
        dataSource: remoteData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs20" %}

In the meantime, the JSON data from the local *.json file type can also be connected to the pivot table via the file uploader option. Here, the resulting string after uploading the file needs to be converted to JSON data that can be assigned to the dataSource property under dataSourceSettings. The following code example illustrates the same.

`javascript

import Vue from "vue";

import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";

import { Uploader } from '@syncfusion/ej2-inputs';

Vue.use(PivotViewPlugin);

export default {

data() {

return {

// Step 1: Initiate the file uploader

let uploadObj: Uploader = new Uploader({

});

uploadObj.appendTo('#fileupload');

let input = document.querySelector('input[type="file"]');

// Step 2: Add the event listener which fires when the *.JSON file is uploaded.

input.addEventListener('change', function (e: Event) {

Copyright © 2001 -2024 Syncfusion Inc.

// Step 3: Initiate the file reader

let reader: FileReader = new FileReader();

reader.onload = function () {

// Step 4: Getting the string output which is to be parsed as JSON.

let result: string[][] = JSON.parse(reader.result as string);

dataSourceSettings: {

// Step 5: The JSON result to be bound as data source.

dataSource: this.result

// Step 6: The appropriate report needs to be provided here.

},

}

reader.readAsText((input as any).files[0]);

});

}

}

}

</script>

< style >

@import "@syncfusion/ej2-vue-pivotview/styles/material.css";

</style>

`

Binding JSON data via remote

In-order to bind remote JSON data, mention the endpoint URL under dataSourceSettings property. The URL property supports both direct downloadable file (*.json) and web service URL.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        url: 'https://cdn.syncfusion.com/data/sales-analysis.json',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        expandAll: false,
        rows: [
          { name: 'EnerType', caption: 'Energy Type' },
        ],
        columns: [
          { name: 'EneSource', caption: 'Energy Source' }
        ],
        values: [
          { name: 'PowUnits', caption: 'Units (GWh)' },
          { name: 'ProCost', caption: 'Cost (MM)' }
        ],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs1" %}

CSV

For CSV data binding, the type property under dataSourceSettings needs to be set as CSV mandatorily.

The CSV format is considered to be the most compact format compared to JSON since it is half the size of JSON. This helps to reduce the bandwidth while transferring to the browser.

Binding CSV data via local

In-order to bind local CSV data to the pivot table, user needs to convert it as string array and then directly assign it to the dataSource property under dataSourceSettings.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { csvdata } from './csvdata.js';
import { isNullOrUndefined } from "@syncfusion/ej2-base";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: this.getCSVData(),
        type: 'CSV',
```

```
        expandAll: false,
        formatSettings: [{ name: 'Total Cost', format: 'C0' }, { name:
'Total Revenue', format: 'C0' }, { name: 'Total Profit', format: 'C0' }],
        drilledMembers: [{ name: 'Item Type', items: ['Baby Food'] }],
        rows: [
            { name: 'Country' },
            { name: 'Region' }
        ],
        columns: [
            { name: 'Sales Channel' },
            { name: 'Item Type' }
        ],
        values: [
            { name: 'Total Cost' },
            { name: 'Total Revenue' },
            { name: 'Total Profit' }
        ],
        filters: []
      },
      height: 350
    }
  }
  methods: {
    getCSVData: function() {
      let dataSource = [];
      let jsonObject = csvdata.split(/\r?\n|\r/);
      for (let i = 0; i < jsonObject.length; i++) {
        if (!isNullOrUndefined(jsonObject[i]) && jsonObject[i] !== "") {
          dataSource.push(jsonObject[i].split(","));
        }
      }
      return dataSource;
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/csvdata-cs1" %}

In the meantime, the CSV data from the local *.csv file type can also be connected to the pivot table via the file uploader option. Here, the resulting string after uploading the file needs to be converted to string array that can be assigned to the dataSource property under dataSourceSettings. The following code example illustrates the same.

`javascript

import Vue from "vue";

import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";

import { Uploader } from '@syncfusion/ej2-inputs';

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(PivotViewPlugin);
export default {
data() {
return {
// Step 1: Initiate the file uploader
let uploadObj: Uploader = new Uploader({
});
uploadObj.appendTo('#fileupload');
let input = document.querySelector('input[type="file"]');
// Step 2: Add the event listener which fires when the *.CSV file is uploaded.
input.addEventListener('change', function (e: Event) {
// Step 3: Initiate the file reader
let reader: FileReader = new FileReader();
reader.onload = function () {
// Step 4: Getting the string output which is to be converted as string[][].
let result: string[][] = (reader.result as string).split('\n').map(function (line) {
return line.split(',');
});
dataSourceSettings: {
// Step 5: The string[][] result to be bound as data source
dataSource: this.result,
type: 'CSV',
// Step 6: The appropriate report needs to be provided here.
},
}
reader.readAsText((input as any).files[0]);
});
}
}
}
</script>
< style >
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
```

</style>
`

Binding CSV data via remote

In-order to bind remote CSV data, mention the endpoint URL under dataSourceSettings property. The URL property supports both direct downloadable file (*.csv) and web service URL.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
         url: "https://bi.syncfusion.com/productservice/api/sales",
        type: "CSV",
        expandAll: false,
        enableSorting: true,
        formatSettings: [
          { name: "Total Cost", format: "C0" },
          { name: "Total Revenue", format: "C0" },
          { name: "Total Profit", format: "C0" }
        ],
        drilledMembers: [{ name: "Item Type", items: ["Baby Food"] }],
        rows: [{ name: "Region" }, { name: "Country" }],
        columns: [{ name: "Item Type" }, { name: "Sales Channel" }],
        values: [
          { name: "Total Cost" },
          { name: "Total Revenue" },
          { name: "Total Profit" }
        ],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs2" %}

Copyright © 2001 -2024 Syncfusion Inc.

Remote Data Binding

To interact with remote data source, provide the endpoint url within DataManager along with appropriate adaptor. By default, DataManager uses ODataAdaptor for remote data-binding.

Binding with OData services

OData is a standardized protocol for creating and consuming data. User can retrieve data from OData service using the DataManager. Refer to the following code example for remote data binding using OData service.

APP.VUE

```
<template>
  <div id="app">
    <ejs-pivotview id="pivotview" :dataSourceSettings="dataSourceSettings"
:height="height"></ejs-pivotview>
  </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, IDataSet } from "@syncfusion/ej2-vue-pivotview";
import { DataManager, WebApiAdaptor, ODataAdaptor, Query, ReturnOption }
from '@syncfusion/ej2-data';
Vue.use(PivotViewPlugin);
let remoteData: DataManager = new DataManager({
  url: "https://bi.syncfusion.com/northwindservice/api/orders",
  adaptor: new WebApiAdaptor(),
  crossDomain: true
});
export default {
  data() {
    return {
      dataSourceSettings: {
        dataSource: remoteData as DataManager,
        expandAll: true,
        filters: [],
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
        formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
        values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs3" %}

Copyright © 2001 -2024 Syncfusion Inc.

Binding with OData V4 services

The OData V4 is an improved version of OData protocols, and the DataManager can be used to retrieve and consume OData V4 services. For more details on OData V4 services, refer to the OData documentation. To bind OData V4 service, use the ODataV4Adaptor.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, IDataSet, FieldList } from "@syncfusion/ej2-vue-
pivotview";
import { DataManager, WebApiAdaptor, ODataV4Adaptor, Query, ReturnOption }
from '@syncfusion/ej2-data';
Vue.use(PivotViewPlugin);
let remoteData: DataManager = new DataManager({
  adaptor: new ODataV4Adaptor,
  url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/?$top=7',
  crossDomain: true
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: remoteData as DataManager,
        expandAll: true,
        filters: [],
        columns: [{ name: 'OrderDate'}, { name: 'ShipCity' }],
        rows: [{ name: 'OrderID' }, { name: 'CustomerID' }],
        values: [{ name: 'Freight' }]
      },
      height: 350,
      showFieldList: true
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs4" %}

Web API

User can use WebApiAdaptor to bind pivot table with Web API created using OData endpoint.

APP.VUE

```
<template>
    <div id="app">
```

```
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, IDataSet } from "@syncfusion/ej2-vue-pivotview";
import { DataManager, WebApiAdaptor, ODataAdaptor, Query, ReturnOption }
from '@syncfusion/ej2-data';
Vue.use(PivotViewPlugin);
let remoteData: DataManager = new DataManager({
  url: "https://bi.syncfusion.com/northwindservice/api/orders",
  adaptor: new WebApiAdaptor(),
  crossDomain: true
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: remoteData as DataManager,
        expandAll: true,
        filters: [],
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
        formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
        values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs5" %}

Querying in Data Manager

By default, the data manager retrieves all the data from the provider which is mapped in it. The data from the provider can be filtered, sorted, paged, etc. by setting the own query in defaultQuery property in the data manager instance.

APP.VUE

```
<template>
  <div id="app">
    <ejs-pivotview id="pivotview" :dataSourceSettings="dataSourceSettings"
:height="height"></ejs-pivotview>
  </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, IDataSet } from "@syncfusion/ej2-vue-pivotview";
```

```
import { DataManager, WebApiAdaptor, ODataAdaptor, Query, ReturnOption }
from '@syncfusion/ej2-data';
Vue.use(PivotViewPlugin);
let remoteData: DataManager = new DataManager({
  url: "https://bi.syncfusion.com/northwindservice/api/orders",
  adaptor: new WebApiAdaptor(),
  crossDomain: true
});
remoteData.defaultQuery = new Query().take(2);
export default {
  data() {
    return {
      dataSourceSettings: {
        dataSource: remoteData as DataManager,
        expandAll: true,
        filters: [],
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
        formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
        values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs6" %}

Mapping

One can define field information like alias name (caption), data type, aggregation type, show and hide subtotals etc. using the fieldMapping property under dataSourceSettings. The available options are,

- name - It is to specify the appropriate field name.
- caption - It is to set the alias name (caption) to the specific field. Instead of actual field name, the alias name (caption) will be set in the UI of the pivot table.
- type - It is to display values in the pivot table with appropriate aggregation such as sum, product, count, average, minimum, maximum, etc. Its default value is **sum**. This option is applicable only for relational data source.
- axis - It will help to display the field in specified axis such as row/column/value/filter axis of the pivot table.
- showNoDataItems - It is to show all the members of a specific field to the pivot table, even if there are no data in the intersection of the row and column. The default value is **false**. This option is applicable only for relational data source.
- baseField - For the aggregate types like "DifferenceFrom" or "PercentageOfDifferenceFrom" or "PercentageOfParentTotal", selective field is assigned for comparison via this property.

Copyright © 2001 -2024 Syncfusion Inc.

- baseItem For the aggregate types like "DifferenceFrom" or "PercentageOfDifferenceFrom" or "PercentageOfParentTotal", selective member in a field is assigned for comparison via this property.
- expandAll - It is to expand or collapse all headers of a specific field in row and column axes of the pivot table. The default value is **false**.
- showSubTotals - It is to show or hide sub-totals of a specific field in row and column axis of the pivot table. The default value is **true**.
- isNamedSet - It is to set whether the specified field is named set or not. In general, the named set is a set of dimension members or a set expression (MDX query) to be created as a dimension in the SSAS OLAP cube itself. The default value is **false** and this option is applicable only for OLAP data source.
- isCalculatedField - It is to set whether the specified field is a calculated field or not. In general, a calculated field is created from the bound data source or using simple formula with basic arithmetic operators in the pivot table. The default value is **false** and this option is applicable only for OLAP data source.
- showFilterIcon - It is to show or hide the filter icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This filter icon is used to filter the members of a specified field at runtime in the pivot table. The default value is **true**.
- showSortIcon - It is to show or hide the sort icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This sort icon is used to order members of a specified field either in ascending or descending at runtime. The default value is **true**.
- showRemoveIcon - It is to show or hide the remove icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This remove icon is used to remove the specified field during runtime. The default value is **true**.
- showValueTypeIcon - It is to show or hide the value type icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This value type icon helps to select the appropriate aggregation type to specified value field at runtime. The default value is **true**.
- showEditIcon - It is to show or hide the edit icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This edit icon is used to modify caption, formula, and format of a specified calculated field at runtime. The default value is **true**.
- allowDragAndDrop - It is to restrict specific field's button from being dragged on runtime in the grouping bar and field list UI. This will prevent from altering the current report. The default value is **true**.
- dataType - It is to specify the type of the field like 'string', 'number', 'datetime', 'date', and 'boolean'.
- groupName - It is to display fields in the field list UI by grouping them under the desired folder name.

The main purpose of these mapping options is to configure each field that is not part of the initial pivot report. Even if any field that is part of this mapping is defined here, the value set in the initial report will have the highest preceding.

This option is applicable only for relational data source.

In the below code sample, visibility of the field button icons are configured.

APP.VUE

```
<template>
    <div id="app">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showGroupingBar="showGroupingBar"
:showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar, FieldList } from "@syncfusion/ej2-
vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        columns: [{ name: 'Year', caption: 'Production Year' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }],
        rows: [{ name: 'Country' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        fieldMapping: [
           { name: 'Quarter', showSortIcon: false },
           { name: 'Products', showFilterIcon: false, showRemoveIcon: false
},
           { name: 'Amount', showValueTypeIcon: false, caption: 'Sold
Amount' },
        ]
      },
      height: 350,
      showGroupingBar: true,
      showFieldList: true,
    }
  },
  provide: {
        pivotview: [GroupingBar, FieldList]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs21" %}

Values in row axis

By default, the value fields are plotted in column axis. To plot those fields in row axis, use the valueAxis property by setting its value as **row**. By default, it holds the value **column**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        valueAxis: 'row'
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs22" %}

Values at different positions

By default, the value fields are placed at the end of the row or column axis. To place those value fields in different positions, use the valueIndex property and set the value to an appropriate index position. Its default value is **-1**, which denotes the last position. The valueIndex property is dependent on the valueAxis property.

This support is only available for relational data sources. Also, enable the showValuesButton property in the grouping bar and field list UI to **true** to re-arrange the values fields at different positions via user interaction.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        valueIndex: 1
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs23" %}

Show 'no data' items

By default, the pivot table only shows the field item if it has data in its row or column combination. To show all items that do not have data in row and column combination in the pivot table, use the showNoDataItems property by settings its value to **true** for the desired fields. In the following code sample, rows of the "County" and "State" fields do not have data in all combination with "Date" column field.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { noData } from './noData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: noData,
        expandAll: true,
```

```
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        columns: [{ name: 'Date', showNoDataItems: true}],
        values: [{ name: 'Quantity', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
        rows: [{ name: 'Country', showNoDataItems: true }, { name: 'State',
showNoDataItems: true }],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/noData-cs1" %}

Show value headers always

To show value header always in pivot table, even if it holds a single value, use the alwaysShowValueHeader property by settings its value as **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        alwaysShowValueHeader: true
      },
      height: 350
    }
  }
}
</script>
```

```
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs24" %}

Customize empty value cells

User can show custom string in empty value cells using the emptyCellsTextContent property in dataSourceSettings of the pivot table. Since the property is of string data type, user can fill empty value cells with any value like "0", "-", "*", "(blank)", etc. Its common for all value fields and can be configured through code behind.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { noData } from './noData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: noData,
        expandAll: true,
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        columns: [{ name: 'Date', showNoDataItems: true}],
        values: [{ name: 'Quantity', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
        rows: [{ name: 'Country', showNoDataItems: true }, { name: 'State',
showNoDataItems: true }],
        filters: [],
        emptyCellsTextContent:'**'
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/noData-cs2" %}

Copyright © 2001 -2024 Syncfusion Inc.

Event

Load

The event load fires before initiate rendering of pivot table. It holds following parameters likedataSourceSettings, fieldsType and pivotView. In this event user can customize data source settings before initiating pivot table render module.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :load="load"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, LoadEventArgs } from "@syncfusion/ej2-vue-
pivotview";
import { noData } from './noData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: noData,
        expandAll: true,
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        columns: [{ name: 'Date'}],
        values: [{ name: 'Quantity', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'State' }],
        filters: [],
        emptyCellsTextContent: '**'
      },
      height: 350
    }
  }
  methods: {
    load:function(args: LoadEventArgs) {
        args.dataSourceSettings.emptyCellsTextContent = "###";
        args.dataSourceSettings.columns[0].caption = "Full Year";
        args.dataSourceSettings.expandAll = false;
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/noData-cs3" %}

EnginePopulated

The event enginePopulated is triggered after engine is populated. It has following parameters - dataSourceSettings, pivotFieldList and pivotValues.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :enginePopulated="enginePopulated"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, EnginePopulatedEventArgs } from "@syncfusion/ej2-
vue-pivotview";
import { noData } from './noData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: noData,
        expandAll: true,
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        columns: [{ name: 'Date'}],
        values: [{ name: 'Quantity', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'State' }],
        filters: [],
        emptyCellsTextContent: '**'
      },
      height: 350
    }
  }
  methods: {
    enginePopulated:function(args: EnginePopulatedEventArgs) {
      //triggers after engine populated
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/noData-cs4" %}

EnginePopulating

The event enginePopulating triggers before the pivot engine starts to populate and allows to customize the pivot datasource settings. It has following parameter dataSourceSettings.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :enginePopulating="enginePopulating"> </ejs-pivotview>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, EnginePopulatingEventArgs  } from
"@syncfusion/ej2-vue-pivotview";
import { noData } from './noData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: noData,
        expandAll: true,
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        columns: [{ name: 'Date'}],
        values: [{ name: 'Quantity', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'State' }],
        filters: [],
        emptyCellsTextContent: '**'
      },
      height: 350
    }
  }
  methods: {
    enginePopulating:function(args: EnginePopulatingEventArgs ) {
       args.dataSourceSettings.columns[0].caption = 'Full Year';
       args.dataSourceSettings.emptyCellsTextContent = '##';
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/noData-cs5" %}

See Also

- Aggregation
- Show/Hide Totals
- Customize number, date, and time values
- Server Side Engine (Optional)

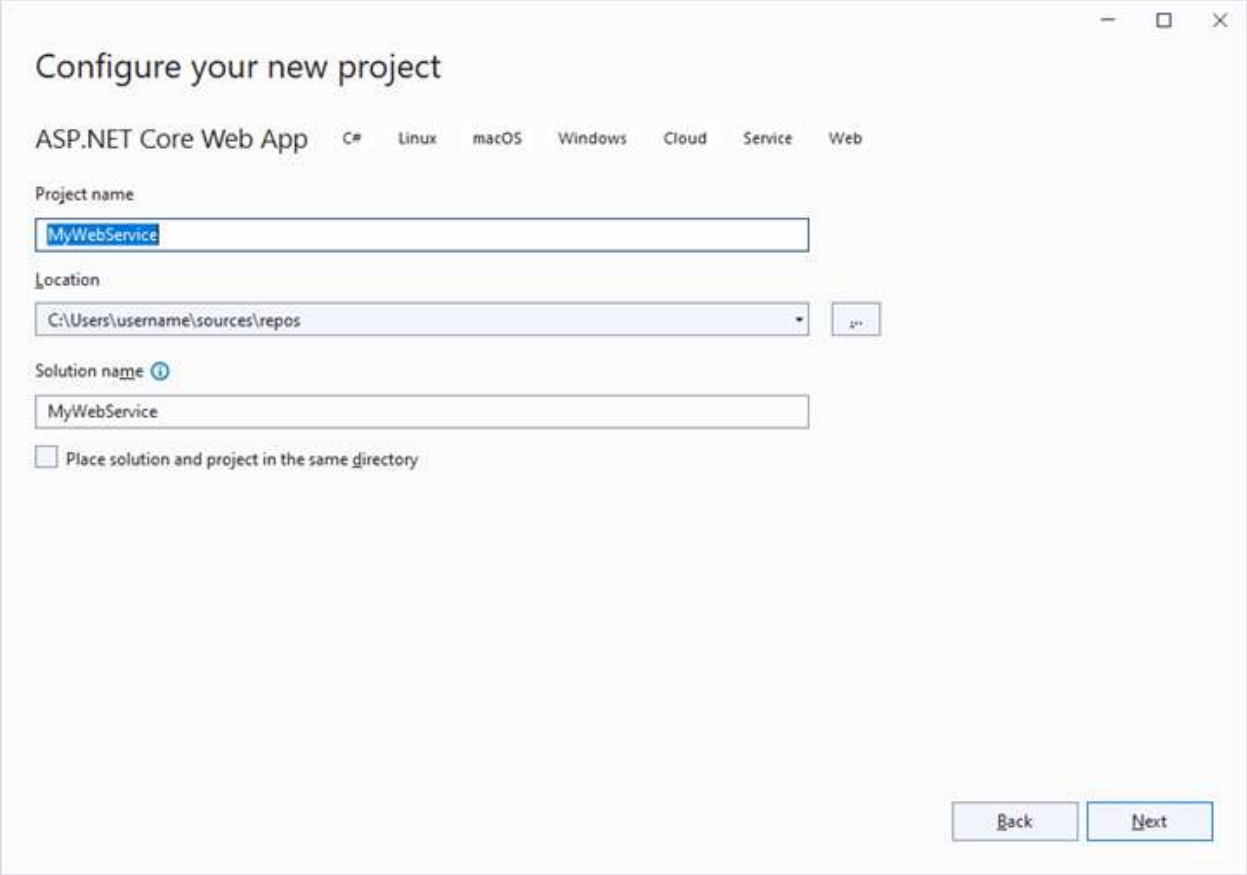
Connecting To Data Source

MySQL in EJ2 Vue Pivotview Component

This section describes how to retrieve data from a MySQL database using MySqlClient and bind it to the Pivot Table via a Web API controller.

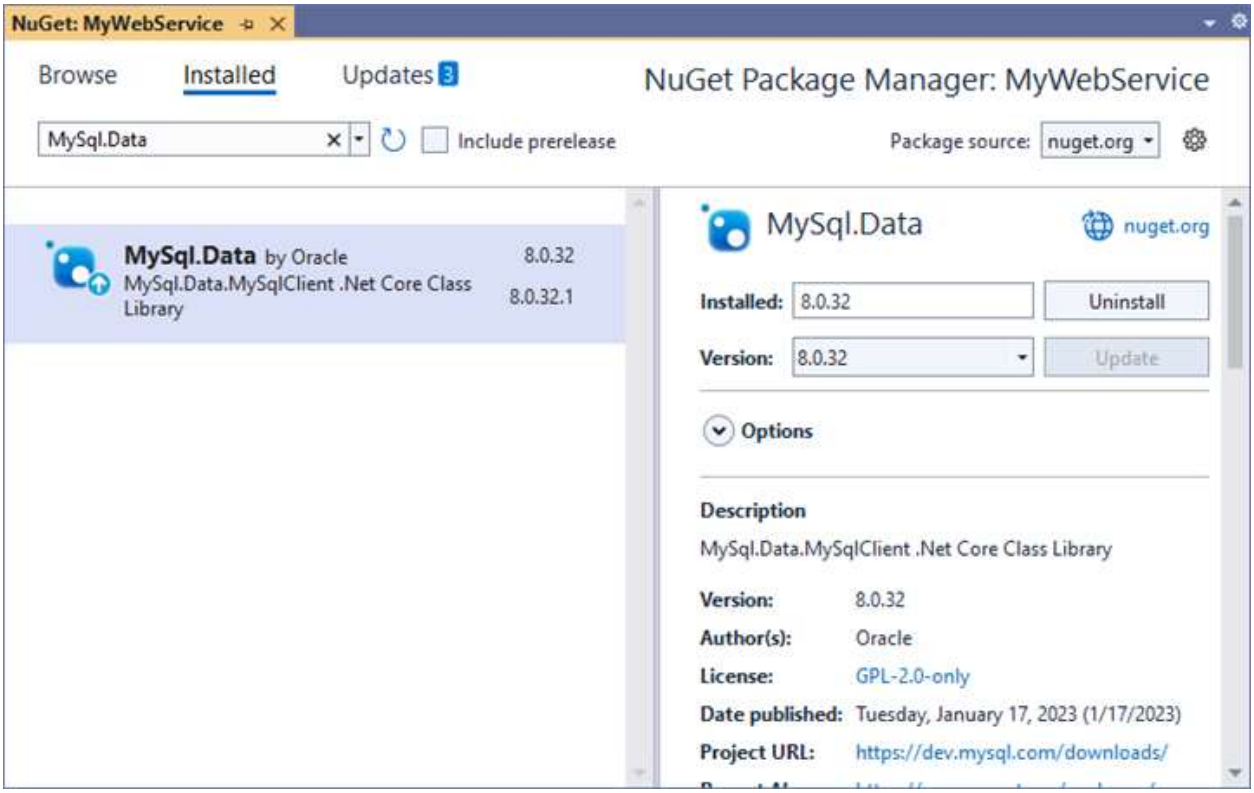
Create a Web API service to fetch MySQL data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document link.



2. To connect a MySQL Server using the **MySqlClient** in our application, we need to install the MySql.Data NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **MySql.Data** and install it.

Copyright © 2001 -2024 Syncfusion Inc.



3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **MySqlConnection** helps to connect the MySQL database. Next, using **MySqlCommand** and **MySqlDataAdapter** you can process the desired query string and retrieve data from the MySQL database. The **Fill** method of the **MySqlDataAdapter** is used to populate the retrieved data into a **DataTable** as shown in the following code snippet.

```csharp
using Microsoft.AspNetCore.Mvc;
using MySql.Data.MySqlClient;
using Newtonsoft.Json;
using System.Data;
namespace MyWebService.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
public dynamic GetMySQLResult()
{
```

```
// Replace with your own connection string.
MySqlConnection connection = new MySqlConnection("<Enter your valid connection string here>");
connection.Open();
MySqlCommand command = new MySqlCommand("SELECT * FROM orders", connection);
MySqlDataAdapter dataAdapter = new MySqlDataAdapter(command);
DataTable dataTable = new DataTable();
dataAdapter.Fill(dataTable);
connection.Close();
return dataTable;
}
}
}
```

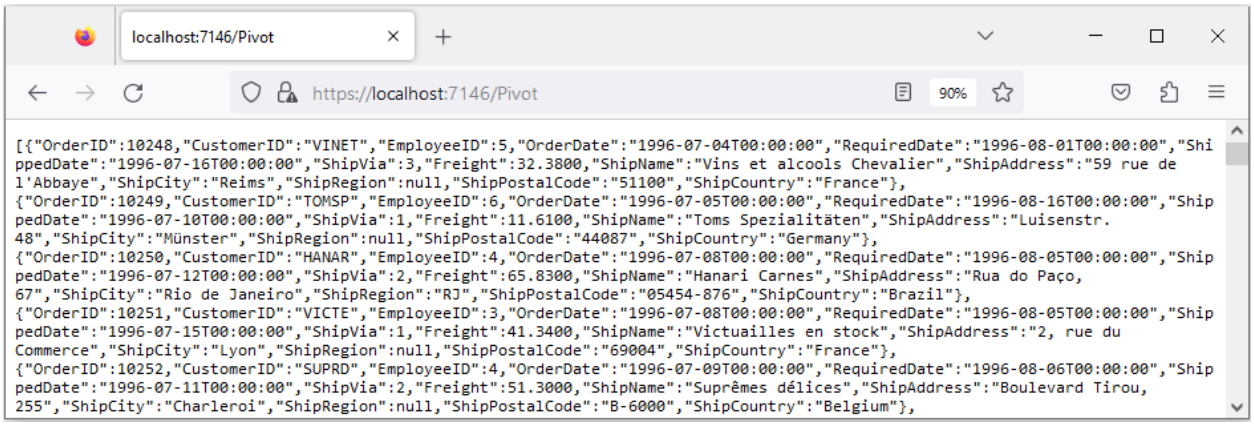
5. In the **Get()** method of the **PivotController.cs** file, the **GetMySQLResult** method is used to retrieve the MySQL data as a **DataTable**, which is then serialized into JSON string using **JsonConvert.SerializeObject()**.

```csharp
using Microsoft.AspNetCore.Mvc;
using MySql.Data.MySqlClient;
using Newtonsoft.Json;
using System.Data;
namespace MyWebService.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpGet(Name = "GetMySQLResult")]
public object Get()
{
return JsonConvert.SerializeObject(GetMySQLResult());
}
public dynamic GetMySQLResult()
{
```

```
// Replace with your own connection string.
MySqlConnection connection = new MySqlConnection("<Enter your valid connection string here>");
connection.Open();
MySqlCommand command = new MySqlCommand("SELECT * FROM orders", connection);
MySqlDataAdapter dataAdapter = new MySqlDataAdapter(command);
DataTable dataTable = new DataTable();
dataAdapter.Fill(dataTable);
connection.Close();
return dataTable;
}
}
}
`
```

6. Run the application and it will be hosted within the URL https://localhost:7146.

7. Finally, the retrieved data from MySQL database which is in the form of JSON can be found in the Web API controller available in the URL link https://localhost:7146/Pivot, as shown in the browser page below.



Connecting the Pivot Table to a MySQL database using the Web API service

1. Create a simple Vue Pivot Table by following the **"Getting Started"** documentation link.

2. Map the hosted Web API's URL link https://localhost:7146/Pivot to the Pivot Table in **App.vue** by using the url property under dataSourceSettings.

```
`ts
<script>
import Vue from "vue";
import {
FieldList,
```

```
PivotViewPlugin,
} from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
url: 'https://localhost:7146/pivot'
//Other codes here...
}
};
}
};
</script>
<style>
//Syncfusion Angular controls styles
</style>
`
```

3. Frame and set the report based on the data retrieved from the MySQL database.

```
`ts
<template>
<div>
<div class="control-section" style="overflow: auto">
<div class="content-wrapper">
<ejs-pivotview
id="pivotview"
ref="pivotview"
:dataSourceSettings="dataSourceSettings"
:showFieldList="showFieldList"
</ejs-pivotview>
</div>
</div>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import {
FieldList,
PivotViewPlugin,
} from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
url: 'https://localhost:7146/Pivot',
columns: [{ name: 'ShipName' }],
values: [{ name: 'Freight', caption: 'Sum of Freight' }],
rows: [{ name: 'ShipCity' }],
filters: []
},
showFieldList: true
};
},
methods: {
},
provide: {
pivotview: [FieldList]
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
`
```

When you run the sample, the resulting pivot table will look like this:

Explore our Vue Pivot Table sample and ASP.NET Core Web Application to extract data from a MySQL database and bind to the Pivot Table in this GitHub repository.

Microsoft SQL server in EJ2 Vue Pivotview Component

This section describes how to retrieve data from SQL Server database using Microsoft SqlClient and bind it to the Pivot Table via a Web API controller.

Steps to connect the SQL Server database via Web API application

1. Download the ASP.NET Core Web Application from this GitHub repository.

2. The application named as **PivotController** (server-side) that is downloaded from the above GitHub repository includes the following files.

- **PivotController.cs** file under **Controllers** folder – This helps to do data communication with Pivot Table.
- **Database1.mdf** file under **App_Data** folder – This MDF (Master Database File) file contains example data.

3. In the Web API controller (aka, PivotController), **SqlConnection** helps to connect the SQL database (that is, Database1.mdf). Next, using **SqlCommand** and **SqlDataAdapter** you can process the desired SQL

Copyright © 2001 -2024 Syncfusion Inc.

query string and retrieve data from the database. The **Fill** method of the DataAdapter is used to populate the SQL data into a **DataTable** as shown in the following code snippet.

```
using Microsoft.AspNetCore.Mvc;
using System.Data;
using System.Data.SqlClient;
namespace PivotController.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
private static DataTable FetchSQLResult()
{
string conSTR = @"<Enter your valid connection string here>";
string xquery = "SELECT * FROM table1";
SqlConnection sqlConnection = new(conSTR);
sqlConnection.Open();
SqlCommand cmd = new(xquery, sqlConnection);
SqlDataAdapter dataAdapter = new(cmd);
DataTable dataTable = new();
dataAdapter.Fill(dataTable);
return dataTable;
}
}
}
```

4. In the **Get()** method of the **PivotController.cs** file, the **FetchSQLResult** method is used to retrieve the SQL data as a **DataTable**, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

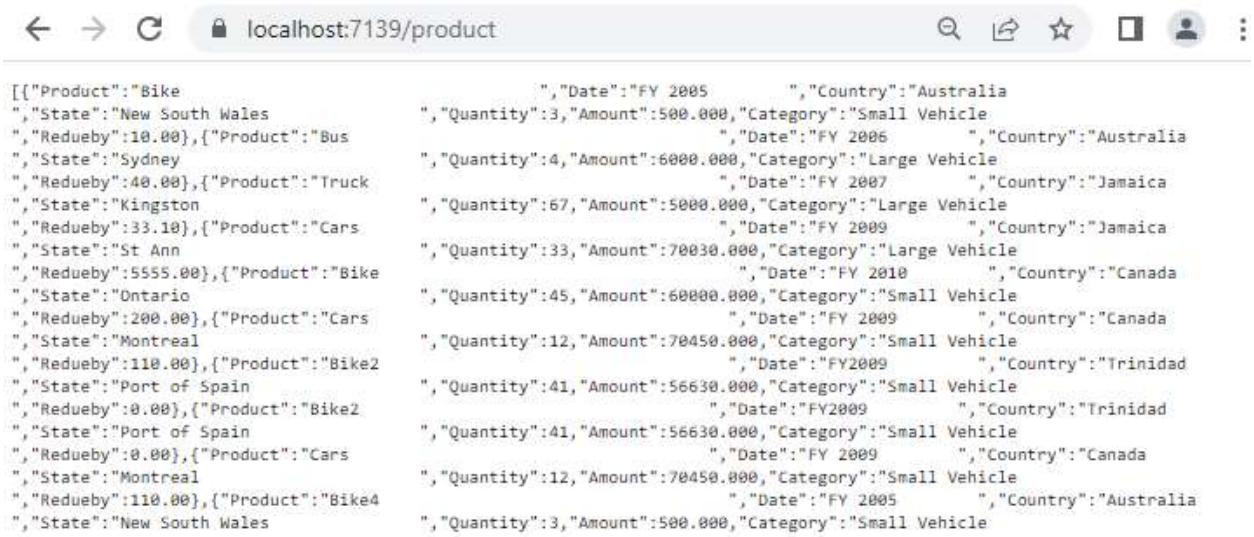
```
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System.Data;
using System.Data.SqlClient;
```

```
namespace PivotController.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpGet(Name = "GetSQLResult")]
public object Get()
{
return JsonConvert.SerializeObject(FetchSQLResult());
}
private static DataTable FetchSQLResult()
{
string conSTR = @"<Enter your valid connection string here>";
string xquery = "SELECT * FROM table1";
SqlConnection sqlConnection = new(conSTR);
sqlConnection.Open();
SqlCommand cmd = new(xquery, sqlConnection);
SqlDataAdapter dataAdapter = new(cmd);
DataTable dataTable = new();
dataAdapter.Fill(dataTable);
return dataTable;
}
}
}
`
```

5. Run the web application (aka, PivotController) and it will be hosted within the URL https://localhost:7139.

6. Finally, the retrieved data from SQL Server which is in the form of JSON can be found in the Web API controller available in the URL link https://localhost:7139/pivot, as shown in the browser page below.

Copyright © 2001 -2024 Syncfusion Inc.



Connecting the Pivot Table to the hosted Web API URL

1. Download the vue Pivot Table sample from this GitHub repository.

2. Next, map the hosted Web API's URL link https://localhost:7139/pivot to the Pivot Table component in **app.ts** by using the url property under dataSourceSettings.

```ts
<script>
import Vue from "vue";
import {
FieldList,
PivotViewPlugin,
} from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
url: 'https://localhost:7139/pivot'
//Other codes here...
}
};
}
};
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
//Syncfusion Angular controls styles
</style>
`
```

3. Frame and set the report based on the data retrieved from the SQL database.

```ts
<template>
<div>
<div class="control-section" style="overflow: auto">
<div class="content-wrapper">
<ejs-pivotview
id="pivotview"
ref="pivotview"
:dataSourceSettings="dataSourceSettings"
:showFieldList="showFieldList"
</ejs-pivotview>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import {
FieldList,
PivotViewPlugin,
} from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
url: 'https://localhost:7139/pivot',
enableSorting: true,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
expandAll: false,
columns: [{ name: 'Product' }],
values: [{ name: 'Quantity' }, { name: 'Amount', caption: 'Sold Amount' }],
rows: [{ name: 'Country' }, { name: 'State' }],
formatSettings: [{ name: 'Amount', format: 'C0' }],
filters: []
},
showFieldList: true
};
},
methods: {
},
provide: {
pivotview: [FieldList]
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
`
```

4. Run the sample to get the following result.

Copyright © 2001 -2024 Syncfusion Inc.

	Large Vehicle		Small Vehicle		Grand Total	
	Quantity	Amount	Quantity	Amount	Quantity	Am
Australia	16	24000	12	2000	28	
Canada			420	1649000	420	
Jamaica	400	300120			400	
Trinidad			328	453040	328	
Grand Total	416	324120	760	2104040	1176	

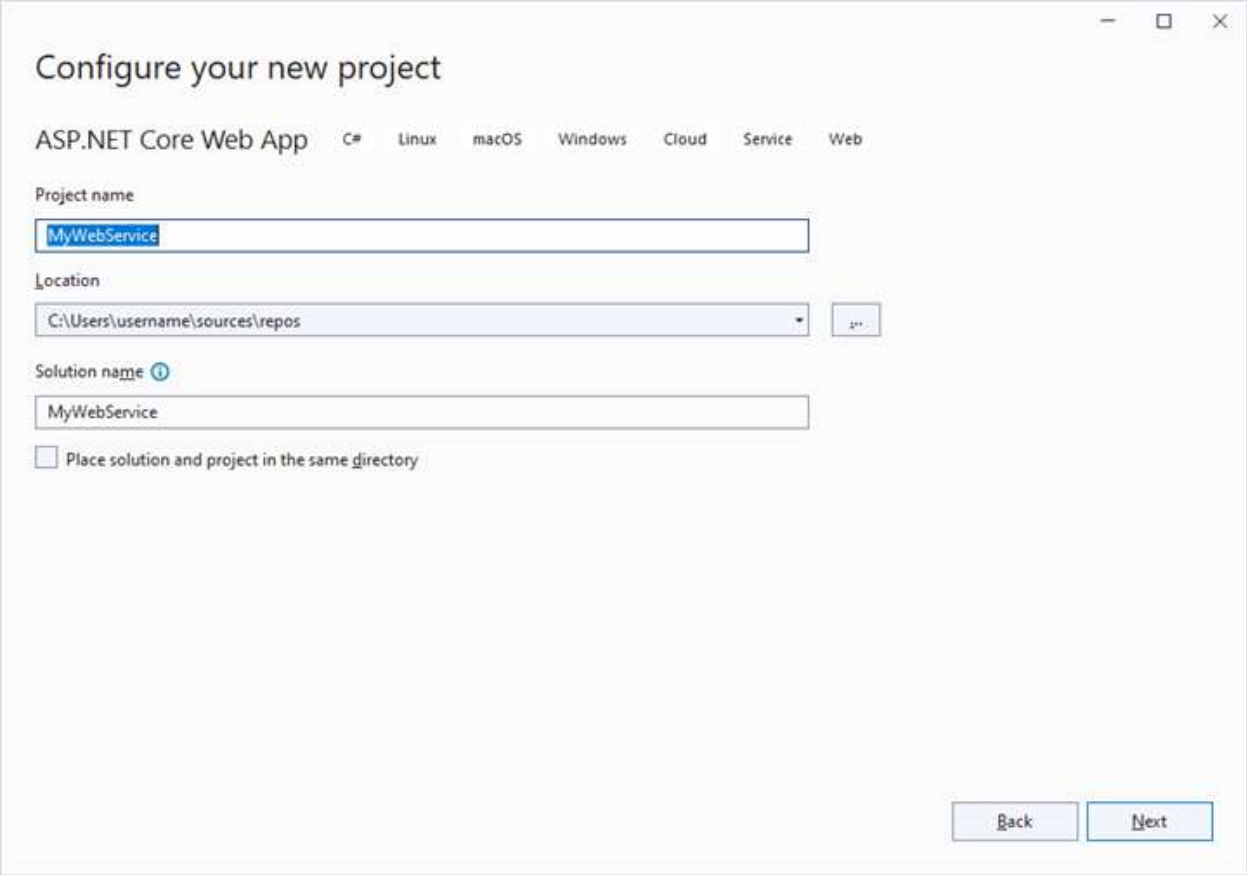
The sample for connecting the Pivot Table to a SQL Server database via an ASP.NET Web application can be found in this GitHub repository.

PostgreSQL in EJ2 Vue Pivotview Component

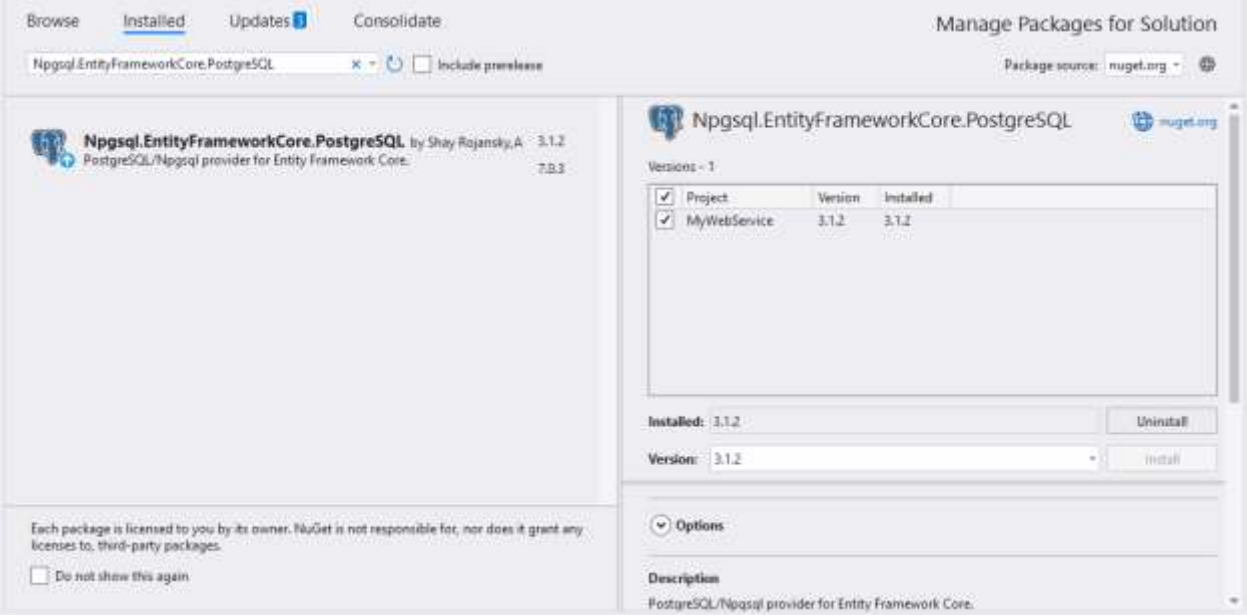
This section describes how to consume data from PostgreSQL database using Microsoft Npgsql and bind it to the Pivot Table via a Web API controller.

Create a Web API service to fetch PostgreSQL data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document link.



2. To connect a PostgreSQL Server using the **Npgsql** in our application, we need to install the Npgsql.EntityFrameworkCore.PostgreSQL NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **Npgsql.EntityFrameworkCore.PostgreSQL** and install it.



3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **NpgsqlConnection** helps to connect the PostgreSQL database. Next, using **NpgsqlCommand** and **NpgsqlDataAdapter** you can process the desired PostgreSQL query string and retrieve data from the database. The **Fill** method of the **NpgsqlDataAdapter** is used to populate the retrieved data into a **DataTable** as shown in the following code snippet.

```csharp
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System.Data;
using Npgsql;
namespace MyWebService.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
public dynamic GetPostgreSQLResult()
{
```

```
// Replace with your own connection string.
NpgsqlConnection connection = new NpgsqlConnection("<Enter your valid connection string here>");
connection.Open();
NpgsqlCommand cmd = new NpgsqlCommand("SELECT * FROM tablename", connection);
NpgsqlDataAdapter da = new NpgsqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
connection.Close();
return dt;
}
}
}
`
```

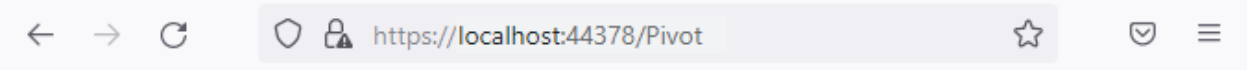
5. In the **Get()** method of the **PivotController.cs** file, the **GetPostgreSQLResult** method is used to retrieve the PostgreSQL data as a **DataTable**, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```csharp
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System.Data;
using Npgsql;
namespace MyWebService.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpGet(Name = "GetPostgreSQLResult")]
public object Get()
{
return JsonConvert.SerializeObject(GetPostgreSQLResult());
}
public dynamic GetPostgreSQLResult()
{
```

```
// Replace with your own connection string.
NpgsqlConnection connection = new NpgsqlConnection("<Enter your valid connection string here>");
connection.Open();
NpgsqlCommand cmd = new NpgsqlCommand("SELECT * FROM tablename", connection);
NpgsqlDataAdapter da = new NpgsqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
connection.Close();
return dt;
}
}
}
`
```

6. Run the application and it will be hosted within the URL https://localhost:44378/.

7. Finally, the retrieved data from PostgreSQL database which is in the form of JSON can be found in the Web API controller available in the URL link https://localhost:44378/Pivot, as shown in the browser page below.




```
[{"revenue":56300,"servicetype":"Retail","servicecategory":"Department","openinghours_practice"
:"08:21:03","closinghours_practice":"05:42:15"},
{"revenue":75400,"servicetype":"Surgical","servicecategory":"Hospital","openinghours_practice":
"10:13:04","closinghours_practice":"03:22:14"},{"revenue":67300,"servicetype":"Non-
Surgical","servicecategory":"Hospital","openinghours_practice":"11:14:06","closinghours_practic
e":"05:26:12"}]
```


Connecting the Pivot Table to a PostgreSQL database using the Web API service

1. Create a simple Vue Pivot Table by following the **"Getting Started"** documentation link.

2. Map the hosted Web API's URL link https://localhost:44378/Pivot to the Pivot Table component in **App.vue** by using the url property under dataSourceSettings.

```
`ts
<script>
import Vue from "vue";
import {
FieldList,
PivotViewPlugin,
} from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
```

```
export default {
data() {
return {
dataSourceSettings: {
url: https://localhost:44378/Pivot
//Other codes here...
}
};
}
};
</script>
<style>
//Syncfusion Angular controls styles
</style>
`
```

3. Frame and set the report based on the data retrieved from the PostgreSQL database.

```
`ts
<template>
<div>
<div class="control-section" style="overflow: auto">
<div class="content-wrapper">
<ejs-pivotview
id="pivotview"
ref="pivotview"
:dataSourceSettings="dataSourceSettings"
:showFieldList="showFieldList"
</ejs-pivotview>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
```

```
import {
FieldList,
PivotViewPlugin,
} from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
url: 'https://localhost:44378/Pivot',
enableSorting: true,
columns: [{ name: 'openinghours practice' }, { name: 'closinghours practice' }],
values: [{ name: 'revenue' }],
rows: [{ name: 'servicetype' }, { name: 'servicecategory' }]
},
showFieldList: true
};
},
methods: {
},
provide: {
pivotview: [FieldList]
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";

@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";

</style>

`

When you run the sample, the resulting pivot table will look like this:

	08:21:03	10:13:04	11:14:06	Grand Total
Non-Surgical			67300	67300
Retail	56300			56300
Surgical		75400		75400
Grand Total	56300	75400	67300	199000

Explore our Vue Pivot Table sample and ASP.NET Core Web Application to extract data from a PostgreSQL database and bind to the Pivot Table in this GitHub repository.

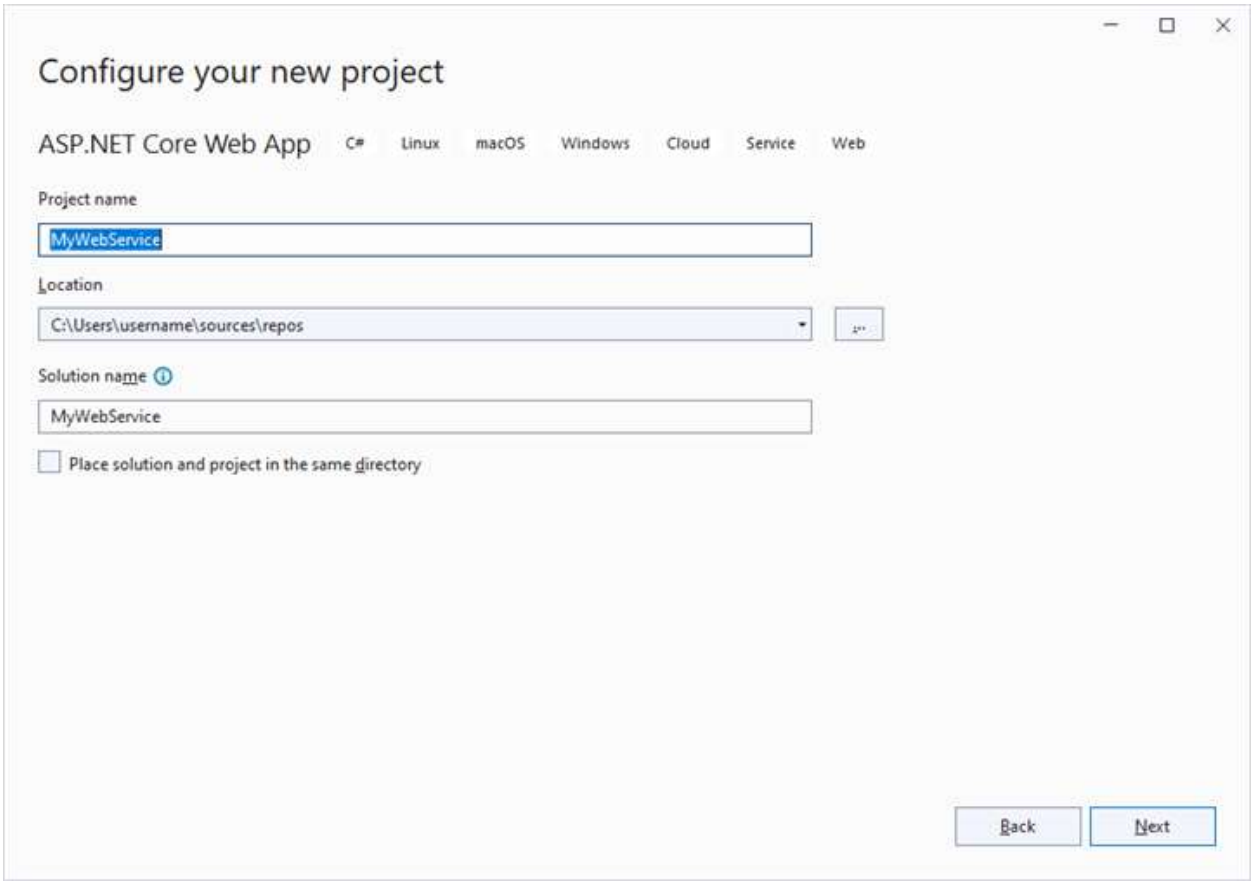
Oracle in EJ2 Vue Pivotview Component

This section describes how to retrieve data from Oracle database using Oracle Managed Data Access and bind it to the Pivot Table via a Web API controller.

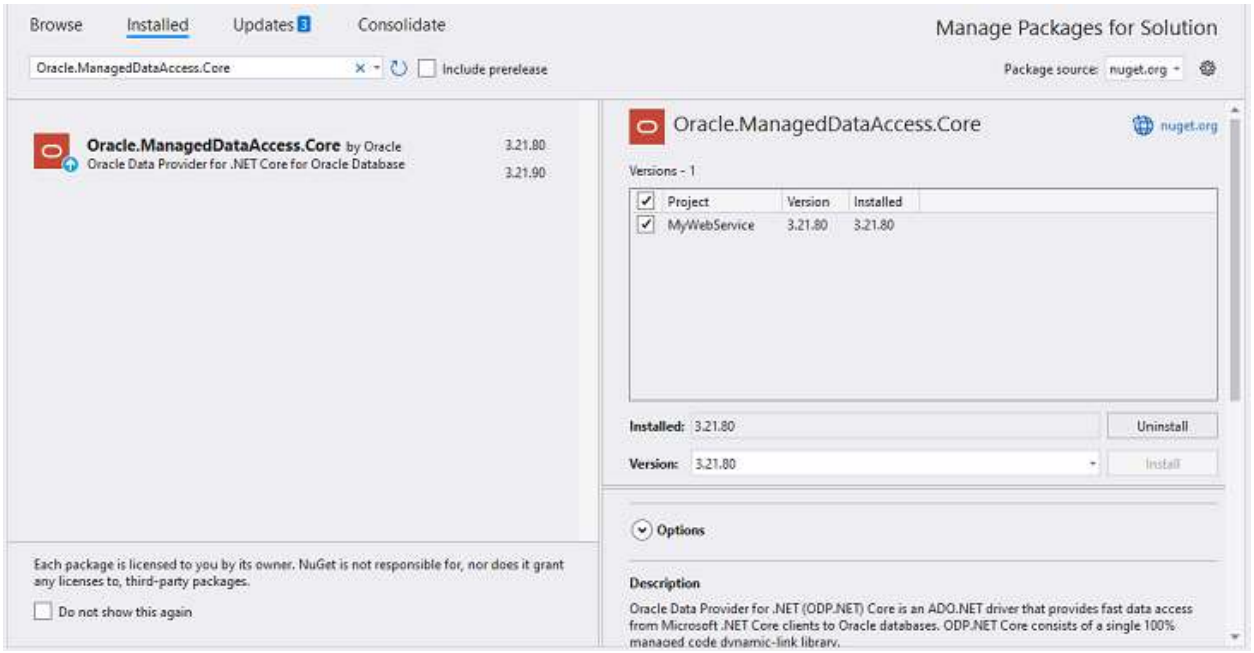
Create a Web API service to fetch Oracle data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document link.

Copyright © 2001 -2024 Syncfusion Inc.



2. To connect a Oracle Server using the **Oracle.ManagedDataAccess.Client** in our application, we need to install the Oracle.ManagedDataAccess.Core NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **Oracle.ManagedDataAccess.Core** and install it.



Copyright © 2001 -2024 Syncfusion Inc.

3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **OracleConnection** helps to connect the Oracle database. Next, using **OracleCommand** and **OracleDataAdapter** you can process the desired Oracle query string and retrieve data from the database. The **Fill** method of the **OracleDataAdapter** is used to populate the retrieved data into a **DataTable** as shown in the following code snippet.

```csharp
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using Oracle.ManagedDataAccess.Client;
using System.Data;
namespace MyWebService.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
private static DataTable FetchOracleResult()
{
// Replace with your own connection string.
string connectionString = "<Enter your valid connection string here>";
OracleConnection oracleConnection = new OracleConnection(connectionString);
oracleConnection.Open();
OracleCommand command = new OracleCommand("SELECT * FROM EMPLOYEES", oracleConnection);
OracleDataAdapter dataAdapter = new OracleDataAdapter(command);
DataTable dataTable = new DataTable();
dataAdapter.Fill(dataTable);
oracleConnection.Close();
return dataTable;
}
}
}
```

5. In the **Get()** method of the **PivotController.cs** file, the **FetchOracleResult()** method is used to retrieve the Oracle data, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

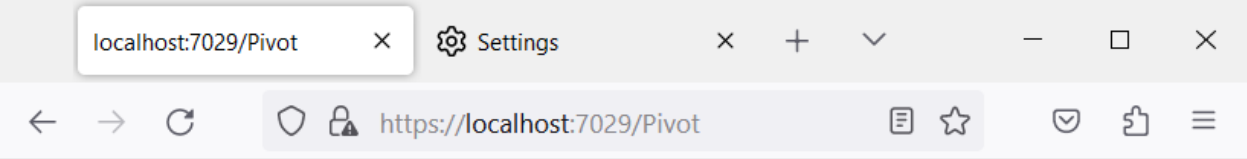
Copyright © 2001 -2024 Syncfusion Inc.

```csharp
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using Oracle.ManagedDataAccess.Client;
using System.Data;
namespace MyWebService.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpGet(Name = "GetOracleResult")]
public object Get()
{
return JsonConvert.SerializeObject(FetchOracleResult());
}
private static DataTable FetchOracleResult()
{
// Replace with your own connection string.
string connectionString = "<Enter your valid connection string here>";
OracleConnection oracleConnection = new OracleConnection(connectionString);
oracleConnection.Open();
OracleCommand command = new OracleCommand("SELECT * FROM EMPLOYEES", oracleConnection);
OracleDataAdapter dataAdapter = new OracleDataAdapter(command);
DataTable dataTable = new DataTable();
dataAdapter.Fill(dataTable);
oracleConnection.Close();
return dataTable;
}
}
}
```

6. Run the application and it will be hosted within the URL https://localhost:44346.

Copyright © 2001 -2024 Syncfusion Inc.

7. Finally, the retrieved data from Oracle database which is in the form of JSON can be found in the Web API controller available in the URL link https://localhost:44346/Pivot, as shown in the browser page below.




```
[{"EMPLOYEE_ID":7369,"EMPLOYEE_NAME":"SMITH","JOB":"CLERK","MANAGER_ID":7902,"HIREDATE":"
1980-12-17T00:00:00","SALARY":800.0,"COMMISSION":null,"DEPARTMENT_ID":20},
{"EMPLOYEE_ID":7499,"EMPLOYEE_NAME":"ALLEN","JOB":"SALESMAN","MANAGER_ID":7698,"HIREDATE"
:"1981-02-20T00:00:00","SALARY":1600.0,"COMMISSION":300.0,"DEPARTMENT_ID":30},
{"EMPLOYEE_ID":7521,"EMPLOYEE_NAME":"WARD","JOB":"SALESMAN","MANAGER_ID":7698,"HIREDATE":
"1981-02-22T00:00:00","SALARY":1250.0,"COMMISSION":500.0,"DEPARTMENT_ID":30},
{"EMPLOYEE_ID":7566,"EMPLOYEE_NAME":"JONES","JOB":"MANAGER","MANAGER_ID":7839,"HIREDATE":
"1981-04-02T00:00:00","SALARY":2975.0,"COMMISSION":null,"DEPARTMENT_ID":20},
{"EMPLOYEE_ID":7654,"EMPLOYEE_NAME":"MARTIN","JOB":"SALESMAN","MANAGER_ID":7698,"HIREDATE
":"1981-09-28T00:00:00","SALARY":1250.0,"COMMISSION":1400.0,"DEPARTMENT_ID":30},
{"EMPLOYEE_ID":7698,"EMPLOYEE_NAME":"BLAKE","JOB":"MANAGER","MANAGER_ID":7839,"HIREDATE":
"1981-05-01T00:00:00","SALARY":2850.0,"COMMISSION":null,"DEPARTMENT_ID":30},
{"EMPLOYEE_ID":7782,"EMPLOYEE_NAME":"CLARK","JOB":"MANAGER","MANAGER_ID":7839,"HIREDATE":
"1981-06-09T00:00:00","SALARY":2450.0,"COMMISSION":null,"DEPARTMENT_ID":10},
{"EMPLOYEE_ID":7788,"EMPLOYEE_NAME":"SCOTT","JOB":"ANALYST","MANAGER_ID":7566,"HIREDATE":
"1987-04-19T00:00:00","SALARY":3000.0,"COMMISSION":null,"DEPARTMENT_ID":20},
{"EMPLOYEE_ID":7839,"EMPLOYEE_NAME":"KING","JOB":"PRESIDENT","MANAGER_ID":null,"HIREDATE"
:"1981-11-17T00:00:00","SALARY":5000.0,"COMMISSION":null,"DEPARTMENT_ID":10},
{"EMPLOYEE_ID":7876,"EMPLOYEE_NAME":"ADAMS","JOB":"CLERK","MANAGER_ID":7788,"HIREDATE":"1
987-05-23T00:00:00","SALARY":1100.0,"COMMISSION":null,"DEPARTMENT_ID":20},
{"EMPLOYEE_ID":7844,"EMPLOYEE_NAME":"TURNER","JOB":"SALESMAN","MANAGER_ID":7698,"HIREDATE
":"1981-09-08T00:00:00","SALARY":1500.0,"COMMISSION":0.0,"DEPARTMENT_ID":30},
{"EMPLOYEE_ID":7900,"EMPLOYEE_NAME":"JAMES","JOB":"CLERK","MANAGER_ID":7698,"HIREDATE":"1
981-12-03T00:00:00","SALARY":950.0,"COMMISSION":null,"DEPARTMENT_ID":30},
{"EMPLOYEE_ID":7902,"EMPLOYEE_NAME":"FORD","JOB":"ANALYST","MANAGER_ID":7566,"HIREDATE":"
1981-12-03T00:00:00","SALARY":3000.0,"COMMISSION":null,"DEPARTMENT_ID":20},
{"EMPLOYEE_ID":7934,"EMPLOYEE_NAME":"MILLER","JOB":"CLERK","MANAGER_ID":7782,"HIREDATE":"
1982-01-23T00:00:00","SALARY":1300.0,"COMMISSION":null,"DEPARTMENT_ID":10}]
```


Connecting the Pivot Table to a Oracle database using the Web API service

1. Create a simple Vue Pivot Table by following the **"Getting Started"** documentation link.

2. Map the hosted Web API's URL link https://localhost:44346/Pivot to the Pivot Table component in **App.vue** by using the url property under dataSourceSettings.

`typescript

<script>

import Vue from "vue";

import {

FieldList,

PivotViewPlugin,

} from "@syncfusion/ej2-vue-pivotview";

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
url: 'https://localhost:44346/Pivot'
//Other codes here...
}
};
}
};
</script>
<style>
//Syncfusion Angular controls styles
</style>
`
```

3. Frame and set the report based on the data retrieved from the Oracle database.

```
`typescript
<template>
<div>
<div class="control-section" style="overflow: auto">
<div class="content-wrapper">
<ejs-pivotview
id="pivotview"
ref="pivotview"
:dataSourceSettings="dataSourceSettings"
:showFieldList="showFieldList"
</ejs-pivotview>
</div>
</div>
</div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import {
FieldList,
PivotViewPlugin,
} from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
url: 'https://localhost:44346/Pivot',
enableSorting: true,
expandAll: false,
columns: [
{ name: 'DEPARTMENT_ID', caption: 'Department ID' },
{ name: 'EMPLOYEE_NAME', caption: 'Employee Name' },
],
rows: [
{ name: 'JOB', caption: 'Job' },
{ name: 'SALARY', caption: 'Salary' }
],
values: [
{ name: 'EMPLOYEE_ID', caption: 'Employee ID' },
{ name: 'CC_EMPLOYEES', caption: 'Employees' },
{ name: 'CCTAXPERCENTAGE', caption: 'Percentage' },
],
filters: []
},
showFieldList: true
};
},
methods: {
},
```

```
provide: {
pivotview: [FieldList]
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
`
```

When you run the sample, the resulting pivot table will look like this:

	10	20	30	Grand Total
ANALYST		15690		15690
CLERK	7934	15245	7900	31079
MANAGER	7782	7566	7698	23046
PRESIDENT	7839			7839
SALESMAN			30518	30518
Grand Total	23555	38501	46116	108172

Explore our Vue Pivot Table sample and ASP.NET Core Web Application to extract data from a Oracle database and bind to the Pivot Table in this GitHub repository.

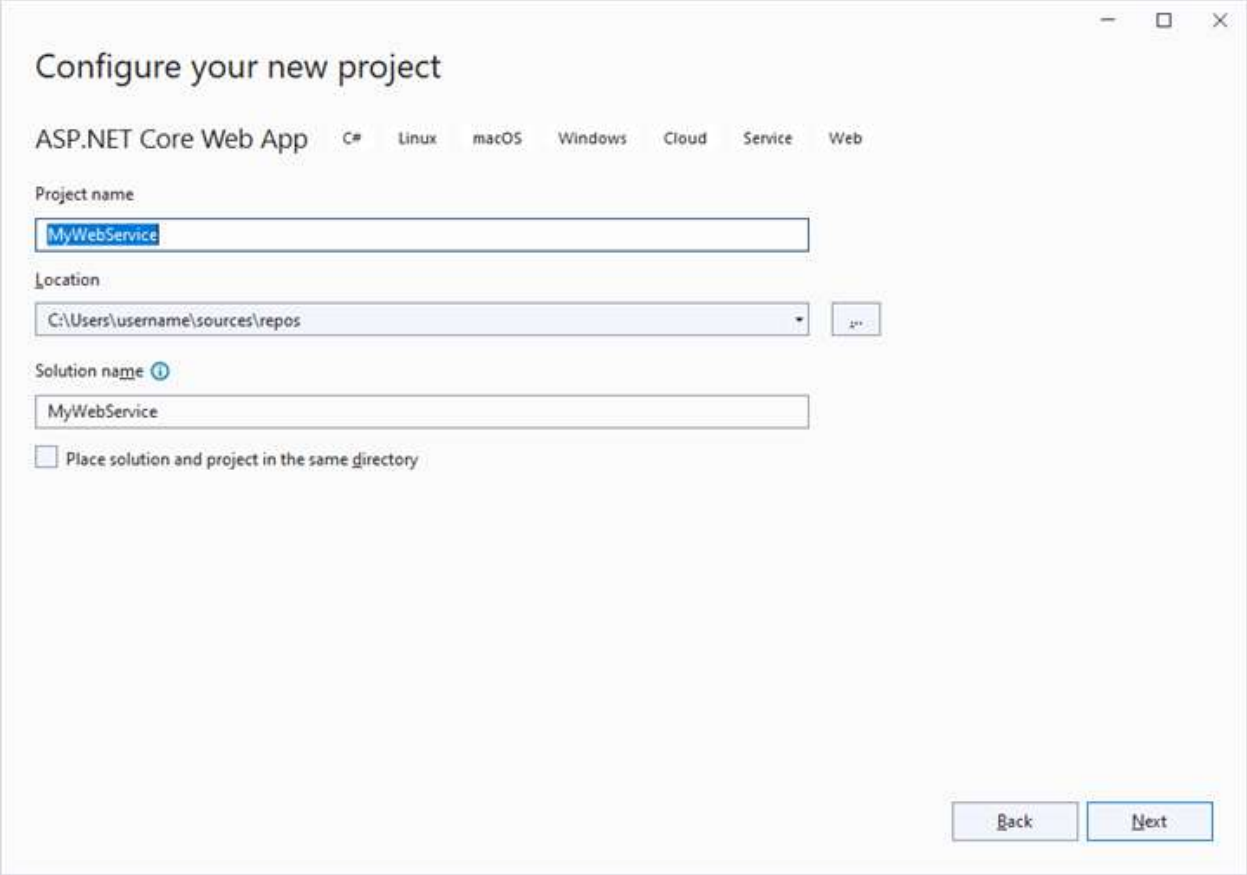
MongoDB in EJ2 Vue Pivotview Component

This section describes how to consume data from MongoDB database using MongoDB Driver and MongoDB Bson libraries and bind it to the Pivot Table via a Web API controller.

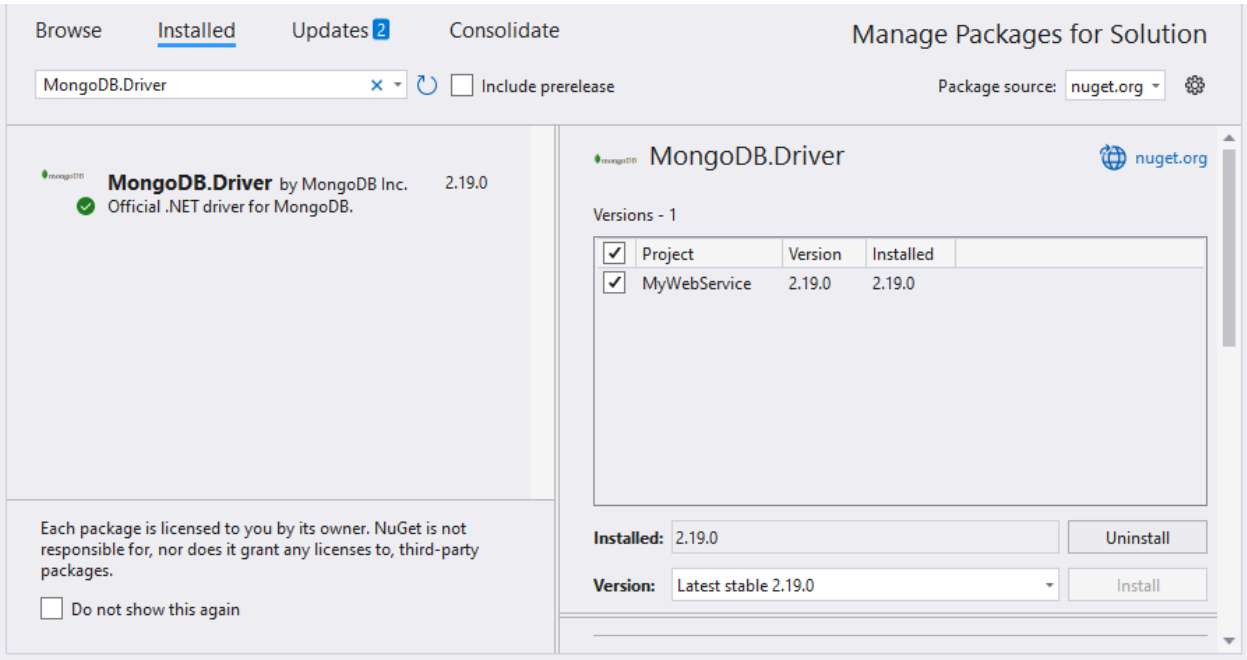
Create a Web API service to fetch MongoDB data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document link.

Copyright © 2001 -2024 Syncfusion Inc.



2. To connect a MongoDB Server using the **MongoDB.Driver** and **MongoDB.Bson** in our application, we need to install the MongoDB.Driver NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **MongoDB.Driver** and install it.



3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **MongoClient** helps to connect the MongoDB database. Next, using the **GetDatabase** and **GetCollection** methods, you can retrieve data from the database. The **Find** method of the **IMongoDatabase** is used to populate the retrieved data into a **List**, as shown in the following code snippet.

```csharp
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using MongoDB.Driver;
using MongoDB.Bson;
namespace MyWebService.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
private static List<ProductDetails> FetchMongoDbResult()
{
// Replace with your own connection string.
string connectionString = "<Enter your valid connection string here>";
MongoClient client = new MongoClient(connectionString);
IMongoDatabase database = client.GetDatabase("sample_training");
var collection = database.GetCollection<ProductDetails>("ProductDetails");
return collection.Find(new BsonDocument()).ToList();
}
public class ProductDetails
{
public ObjectId Id { get; set; }
public int Sold { get; set; }
public double Amount { get; set; }
public string? Country { get; set; }
public string? Products { get; set; }
public string? Year { get; set; }
```

```
public string? Quarter { get; set; }
}
}
}
```

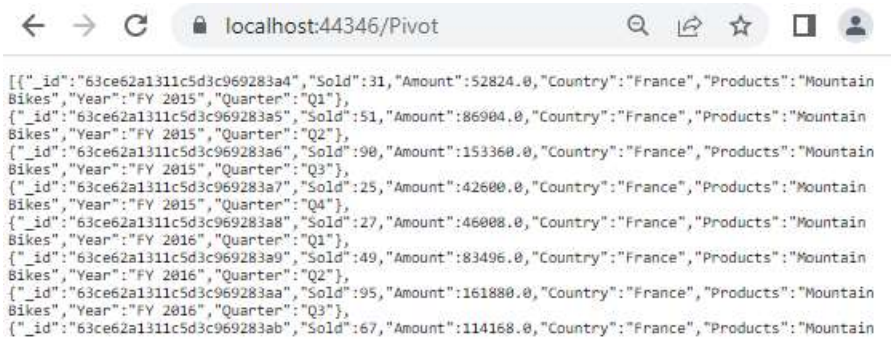
5. In the **Get()** method of the **PivotController.cs** file, the **FetchMongoDbResult()** method is used to retrieve the MongoDB data as a **List**, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```csharp
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using MongoDB.Bson;
using MongoDB.Driver;
namespace MyWebService.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpGet(Name = "GetMongoDbResult")]
public object Get()
{
return JsonConvert.SerializeObject(FetchMongoDbResult());
}
private static List<ProductDetails> FetchMongoDbResult()
{
// Replace with your own connection string.
string connectionString = "<Enter your valid connection string here>";
MongoClient client = new MongoClient(connectionString);
IMongoDatabase database = client.GetDatabase("sample_training");
var collection = database.GetCollection<ProductDetails>("ProductDetails");
return collection.Find(new BsonDocument()).ToList();
}
public class ProductDetails
```

```
{
public ObjectId Id { get; set; }
public int Sold { get; set; }
public double Amount { get; set; }
public string? Country { get; set; }
public string? Products { get; set; }
public string? Year { get; set; }
public string? Quarter { get; set; }
}
}
}
`
```

6. Run the web application and it will be hosted within the URL https://localhost:44346/.

7. Finally, the retrieved data from MongoDB database which is in the form of JSON can be found in the Web API controller available in the URL link https://localhost:44346/Pivot, as shown in the browser page below.



Connecting the Pivot Table to a MongoDB database using the Web API service

1. Create a simple Vue Pivot Table by following the **"Getting Started"** documentation link.

2. Map the hosted Web API's URL link https://localhost:44346/Pivot to the Pivot Table component in **App.vue** by using the url property under dataSourceSettings.

```
`ts
<script>
import Vue from "vue";
import {
FieldList,
PivotViewPlugin,
} from "@syncfusion/ej2-vue-pivotview";
```

```
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
url: 'https://localhost:44346/Pivot',
//Other codes here...
}
};
}
};
</script>
<style>
//Syncfusion Angular controls styles
</style>
`
```

3. Frame and set the report based on the data retrieved from the MongoDB database.

```
`ts
<template>
<div>
<div class="control-section" style="overflow: auto">
<div class="content-wrapper">
<ejs-pivotview
id="pivotview"
ref="pivotview"
:dataSourceSettings="dataSourceSettings"
:showFieldList="showFieldList"
</ejs-pivotview>
</div>
</div>
</div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import {
FieldList,
PivotViewPlugin,
} from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
url: 'https://localhost:44346/Pivot',
enableSorting: true,
columns: [
{ name: 'Year' }
],
values: [
{ name: 'Sold', caption: "Units Sold"},
{ name: 'Amount', caption: "Sold Amount"}
],
rows: [
{ name: 'Country' },
{ name: 'Products' }
]
},
showFieldList: true
};
},
methods: {
},
provide: {
pivotview: [FieldList]
}
};
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
`
```

When you run the sample, the resulting pivot table will look like this:

	FY 2015		FY 2016		F
	Units Sold	Sold Amount	Units Sold	Sold Amount	U
France	729	1160099.5	609	983317	
Germany	528	845472	667	1067220	
United Kingdom	782	1263109.5	640	1031630.5	
United States	682	1085398.5	480	770362	
Grand Total	2721	4354079.5	2396	3852529.5	

Explore our Vue Pivot Table sample and ASP.NET Core Web Application to extract data from a MongoDB database and bind to the Pivot Table in this GitHub repository.

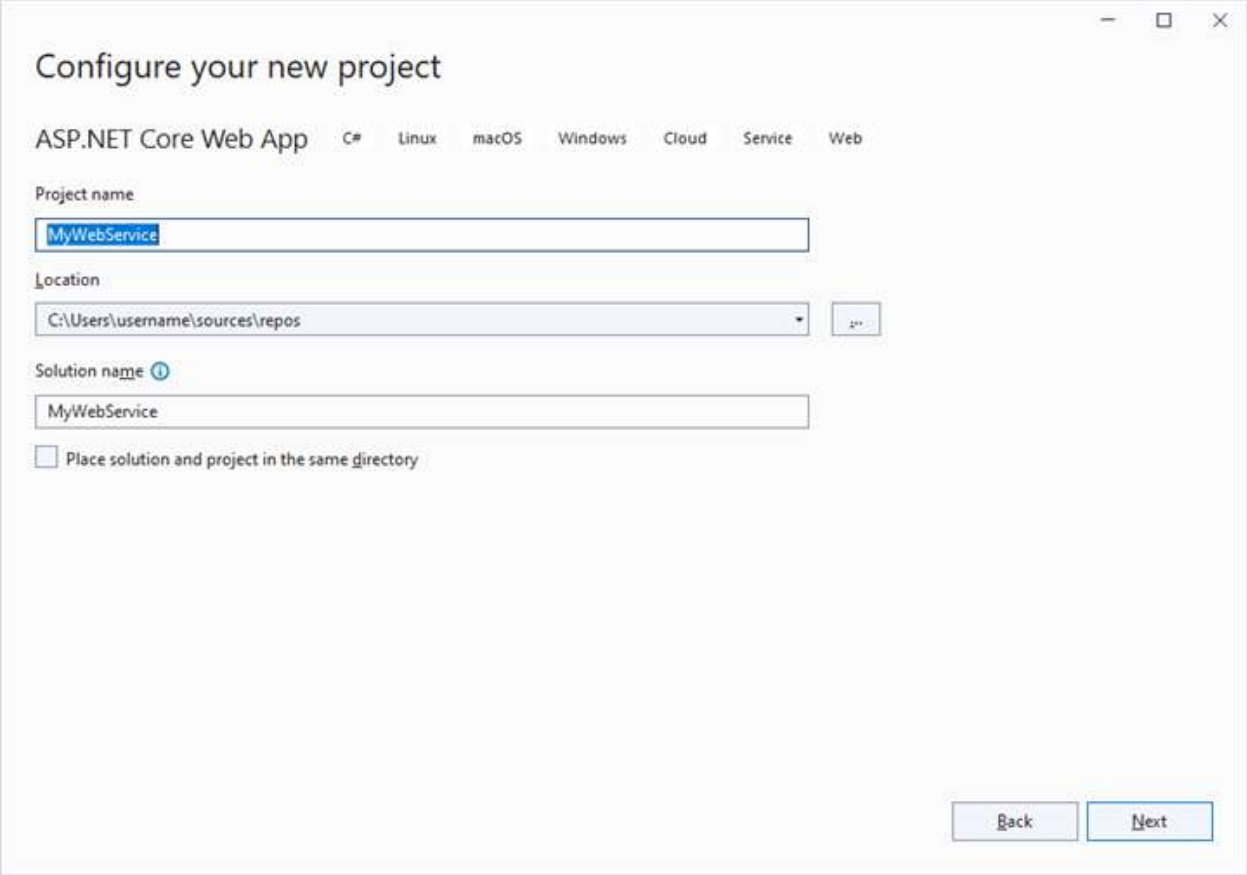
Elasticsearch in EJ2 Vue Pivotview Component

This section describes how to retrieve data from Elasticsearch database using Nest library and bind it to the Pivot Table via a Web API controller.

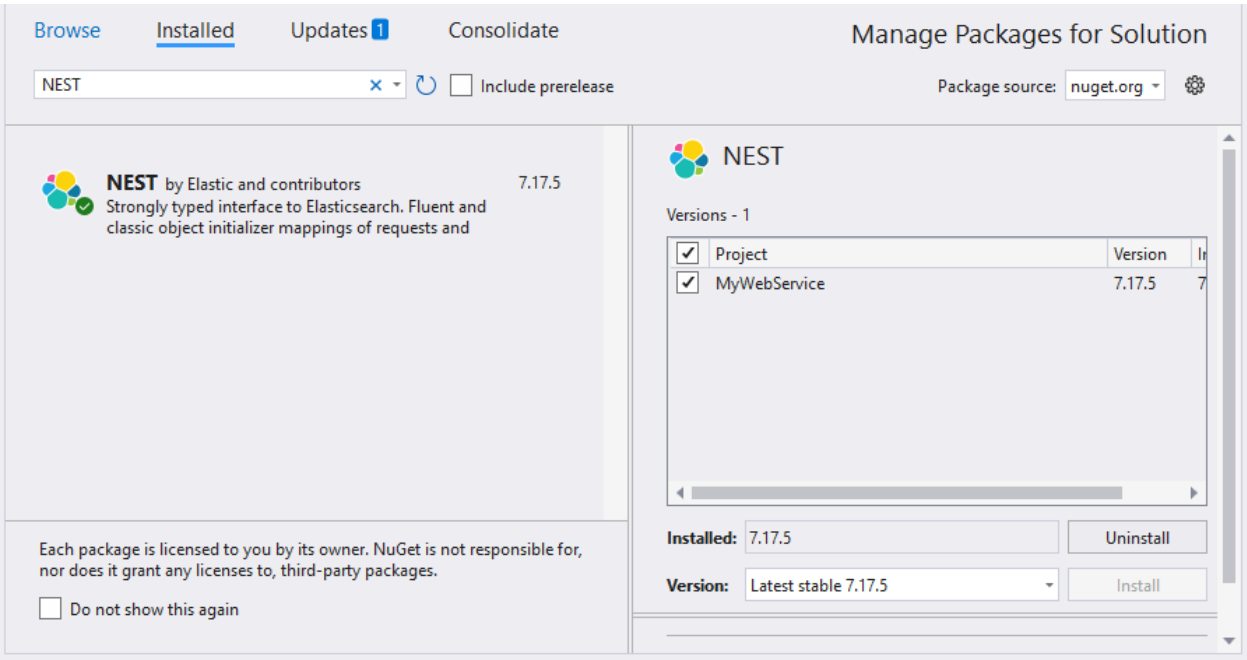
Create a Web API service to fetch Elasticsearch data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document link.

Copyright © 2001 -2024 Syncfusion Inc.



2. To connect a Elasticsearch Server using the **NEST** in our application, we need to install the NEST NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **NEST** and install it.



3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **ElasticClient** helps to connect the Elasticsearch database. Next, using **Search** method you can query your Elasticsearch index and retrieve results from the database.

5. In the **Get()** method of the **PivotController.cs** file, the **FetchElasticsearchData** method is used to retrieve the Elasticsearch data, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```csharp
using Microsoft.AspNetCore.Mvc;
using Nest;
using Newtonsoft.Json;
namespace MyWebService.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpGet(Name = "GetElasticSearchData")]
public object Get()
{
return JsonConvert.SerializeObject(FetchElasticsearchData());
}
private static object FetchElasticsearchData()
{
// Replace with your own connection string.
var connectionString = "<Enter your valid connection string here>";
var uri = new Uri(connectionString);
var connectionSettings = new ConnectionSettings(uri);
var client = new ElasticClient(connectionSettings);
var searchResponse = client.Search<object>(s => s
.Index("product")
.Size(1000)
);
return searchResponse.Documents;
```

Copyright © 2001 -2024 Syncfusion Inc.

}

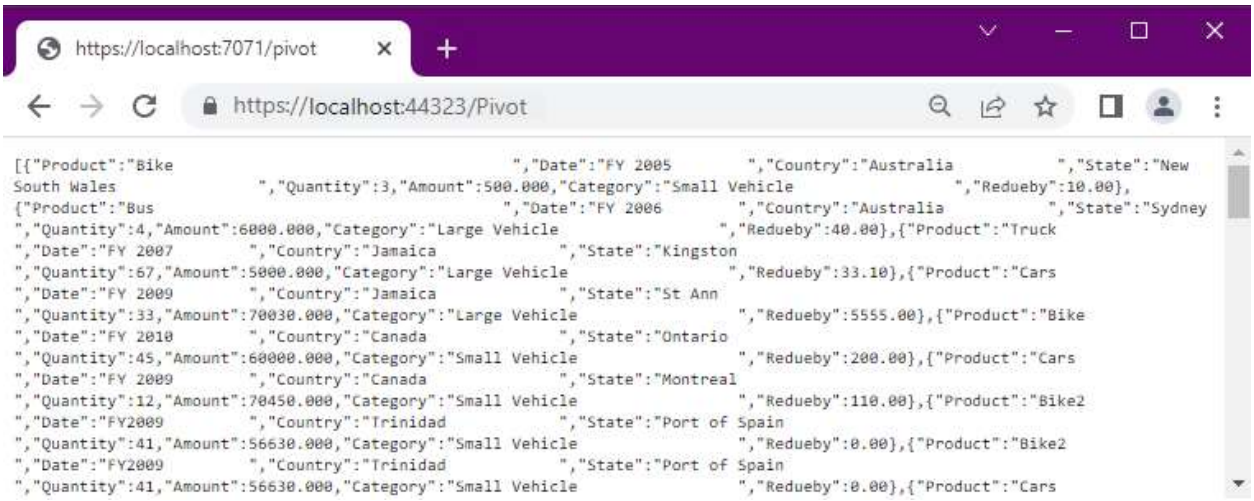
}

}

`

6. Run the web application and it will be hosted within the URL https://localhost:44323.

7. Finally, the retrieved data from Elasticsearch database which is in the form of JSON can be found in the Web API controller available in the URL link https://localhost:44323/Pivot, as shown in the browser page below.



Connecting the Pivot Table to a Elasticsearch database using the Web API service

1. Create a simple Vue Pivot Table by following the **"Getting Started"** documentation link.

2. Map the hosted Web API's URL link https://localhost:44323/Pivot to the Pivot Table component in **App.vue** by using the url property under dataSourceSettings.

`typescript

<script>

import Vue from "vue";

import {

FieldList,

PivotViewPlugin,

} from "@syncfusion/ej2-vue-pivotview";

Vue.use(PivotViewPlugin);

export default {

data() {

return {

dataSourceSettings: {

```
url: 'https://localhost:44323/Pivot'
//Other codes here...
}
};
}
};
</script>
<style>
//Syncfusion Angular controls styles
</style>
`
```

3. Frame and set the report based on the data retrieved from the Elasticsearch database.

```
`typescript
<template>
<div>
<div class="control-section" style="overflow: auto">
<div class="content-wrapper">
<ejs-pivotview
id="pivotview"
ref="pivotview"
:dataSourceSettings="dataSourceSettings"
:showFieldList="showFieldList"
</ejs-pivotview>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import {
FieldList,
PivotViewPlugin,
} from "@syncfusion/ej2-vue-pivotview";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
url: 'https://localhost:44323/Pivot',
expandAll: false,
enableSorting: true,
columns: [{ name: 'Product' }],
values: [{ name: 'Quantity' }, { name: 'Amount', caption: 'Sold Amount' }],
rows: [{ name: 'Country' }, { name: 'State' }],
formatSettings: [{ name: 'Amount', format: 'C0' }],
filters: []
},
showFieldList: true
};
},
methods: {
},
provide: {
pivotview: [FieldList]
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

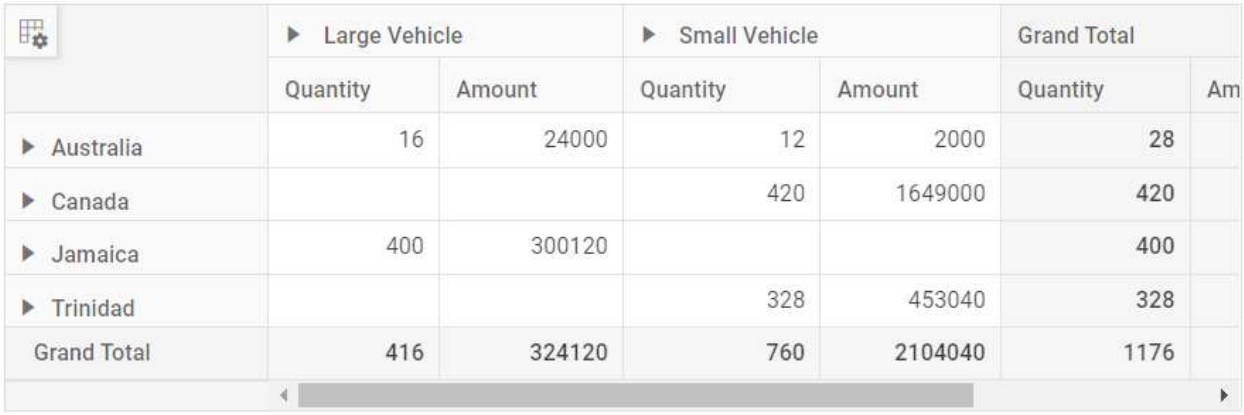
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";

@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";

</style>

`

When you run the sample, the resulting pivot table will look like this:



	Large Vehicle		Small Vehicle		Grand Total	
	Quantity	Amount	Quantity	Amount	Quantity	Am
Australia	16	24000	12	2000	28	
Canada			420	1649000	420	
Jamaica	400	300120			400	
Trinidad			328	453040	328	
Grand Total	416	324120	760	2104040	1176	

Explore our Vue Pivot Table sample and ASP.NET Core Web Application to extract data from a Elasticsearch database and bind to the Pivot Table in this GitHub repository.

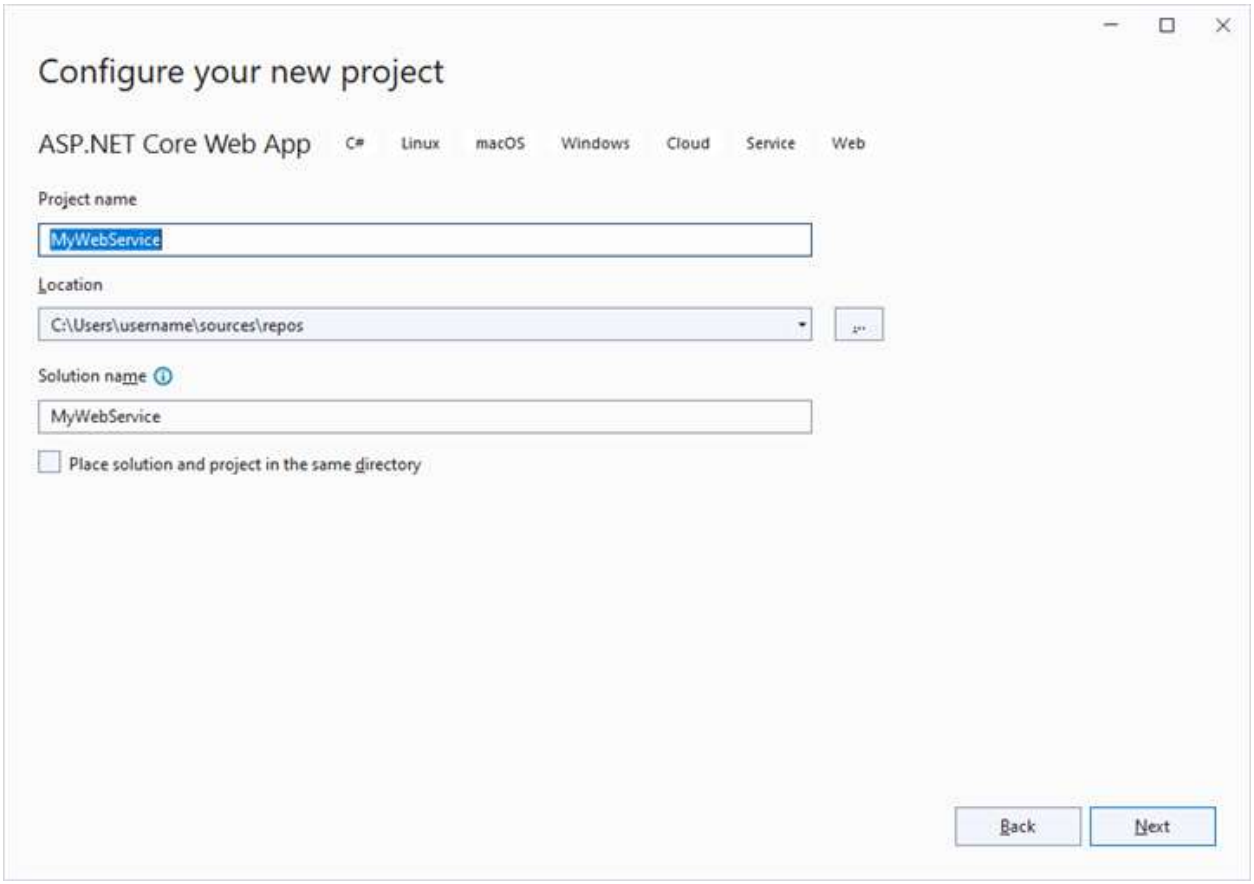
Snowflake in EJ2 Vue Pivotview Component

This section describes how to retrieve data from a Snowflake database using Snowflake Data and bind it to the Pivot Table via a Web API controller.

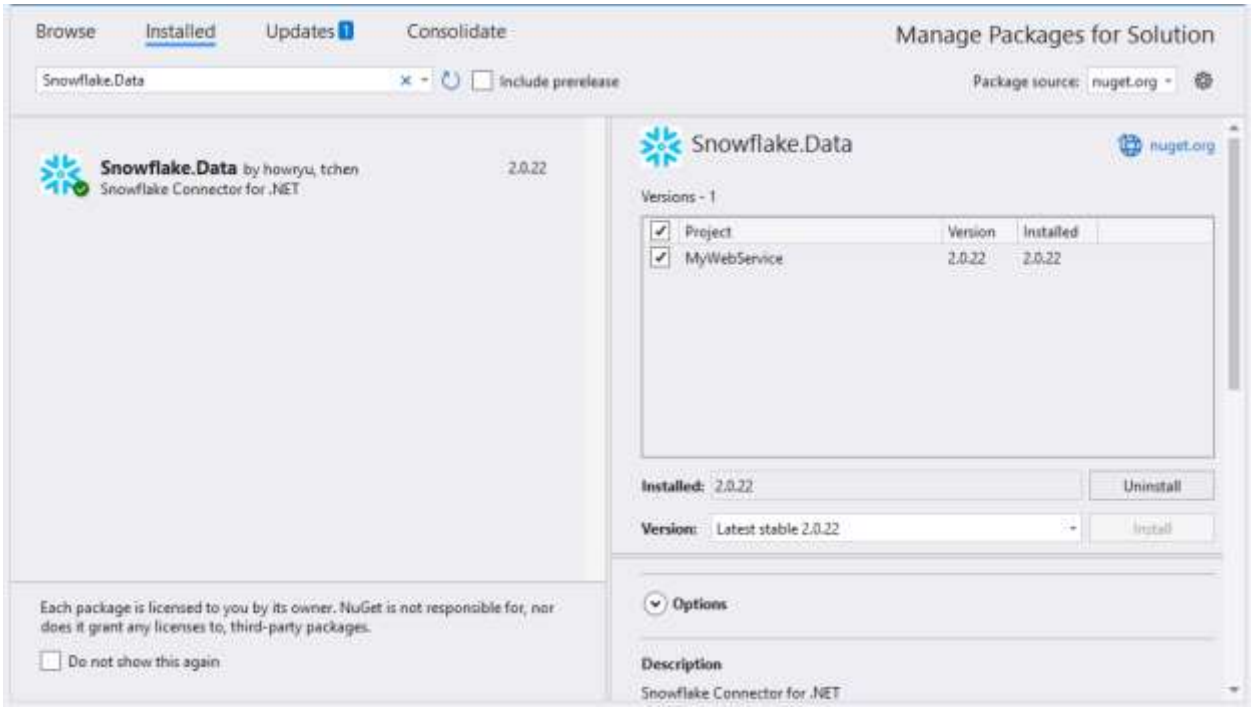
Create a Web API service to fetch Snowflake data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document link.

Copyright © 2001 -2024 Syncfusion Inc.



2. To connect a Snowflake Server using the **Snowflake.Data.Client** in our application, we need to install the Snowflake.Data NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **Snowflake.Data** and install it.



Copyright © 2001 -2024 Syncfusion Inc.

3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **SnowflakeDbConnection** helps to connect the Snowflake database. Next, using **SnowflakeDbDataAdapter** you can process the desired Snowflake query string and retrieve data from the database. The **Fill** method of the **SnowflakeDbDataAdapter** is used to populate the retrieved data into a **DataTable** as shown in the following code snippet.

```csharp
using Microsoft.AspNetCore.Mvc;
using Snowflake.Data.Client;
using Newtonsoft.Json;
using System.Data;
namespace MyWebService.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpGet(Name = "GetSnowflakeResult")]
public object Get()
{
return JsonConvert.SerializeObject(FetchSnowflakeResult());
}
public static DataTable FetchSnowflakeResult()
{
using (SnowflakeDbConnection snowflakeConnection = new SnowflakeDbConnection())
{
// Replace with your own connection string.
snowflakeConnection.ConnectionString = "<Enter your valid connection string here>";
snowflakeConnection.Open();
SnowflakeDbDataAdapter adapter = new SnowflakeDbDataAdapter("select * from CALL_CENTER", snowflakeConnection);
DataTable dataTable = new DataTable();
adapter.Fill(dataTable);
snowflakeConnection.Close();
return dataTable;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
}
}
}
`
```

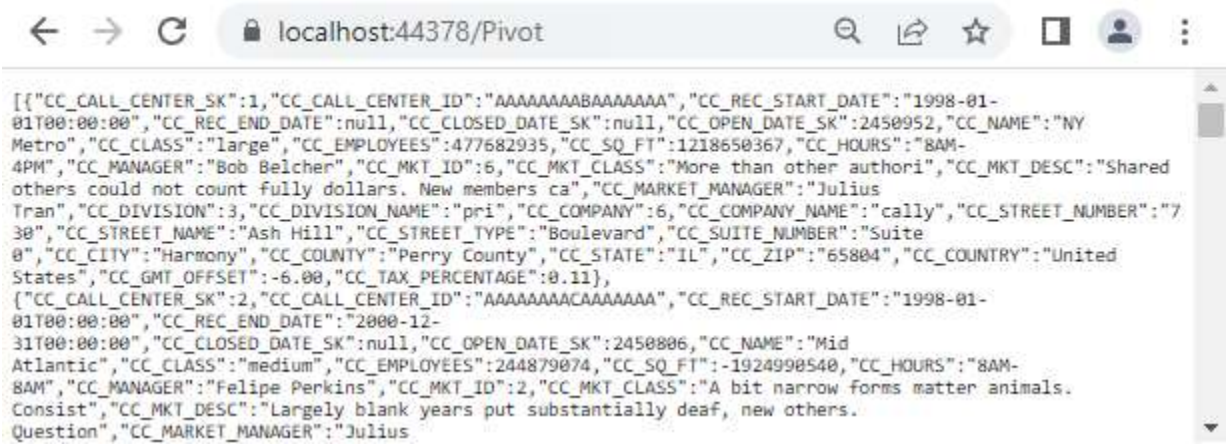
5. In the **Get()** method of the **PivotController.cs** file, the **FetchSnowflakeResult** method is used to retrieve the Snowflake data as a **DataTable**, which is then serialized into JSON string using **JsonConvert.SerializeObject()**.

```csharp
using Microsoft.AspNetCore.Mvc;
using Snowflake.Data.Client;
using Newtonsoft.Json;
using System.Data;
namespace MyWebService.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpGet(Name = "GetSnowflakeResult")]
public object Get()
{
return JsonConvert.SerializeObject(FetchSnowflakeResult());
}
public static DataTable FetchSnowflakeResult()
{
using (SnowflakeDbConnection snowflakeConnection = new SnowflakeDbConnection())
{
// Replace with your own connection string.
snowflakeConnection.ConnectionString = "<Enter your valid connection string here>";
snowflakeConnection.Open();
SnowflakeDbDataAdapter adapter = new SnowflakeDbDataAdapter("select * from CALL_CENTER", snowflakeConnection);
DataTable dataTable = new DataTable();
```

```
adapter.Fill(dataTable);
snowflakeConnection.Close();
return dataTable;
}
}
}
}
`
```

6. Run the application and it will be hosted within the URL https://localhost:44378/.

7. Finally, the retrieved data from Snowflake database which is in the form of JSON can be found in the Web API controller available in the URL link https://localhost:44378/Pivot, as shown in the browser page below.



Connecting the Pivot Table to a Snowflake database using the Web API service

1. Create a simple Vue Pivot Table by following the **"Getting Started"** documentation link.

2. Map the hosted Web API's URL link https://localhost:44378/Pivot to the Pivot Table component in **App.vue** by using the url property under dataSourceSettings.

```
`ts
<script>
import Vue from "vue";
import {
FieldList,
PivotViewPlugin,
} from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
```

```
data() {
return {
dataSourceSettings: {
url: 'https://localhost:44378/pivot'
//Other codes here...
}
};
}
};
</script>
<style>
//Syncfusion Angular controls styles
</style>
`
```

3. Frame and set the report based on the data retrieved from the Snowflake database.

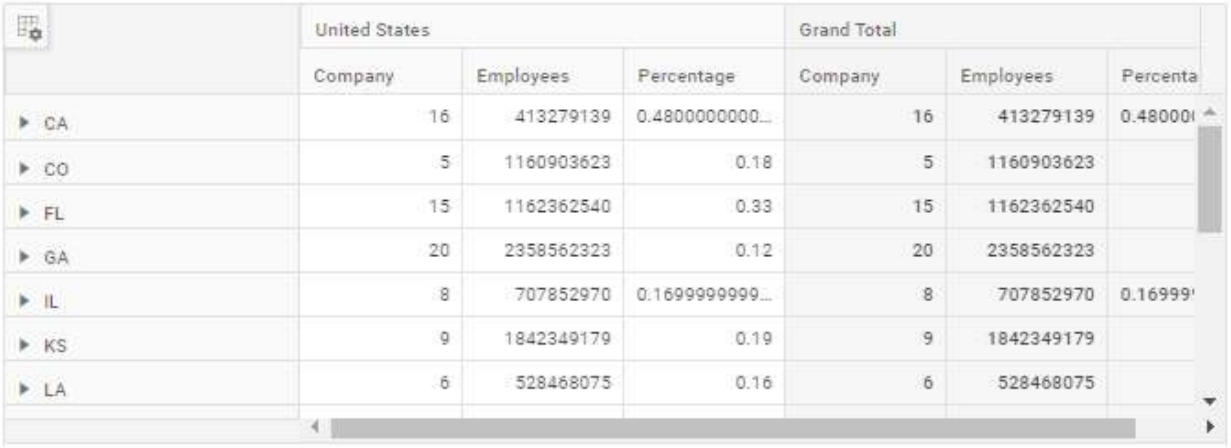
```
`ts
<template>
<div>
<div class="control-section" style="overflow: auto">
<div class="content-wrapper">
<ejs-pivotview
id="pivotview"
ref="pivotview"
:dataSourceSettings="dataSourceSettings"
:showFieldList="showFieldList"
</ejs-pivotview>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import {
```

```
FieldList,
PivotViewPlugin,
} from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
url: 'https://localhost:44378/Pivot',
enableSorting: true,
expandAll: false,
dataSource: [],
columns: [
{ name: 'CC_COUNTRY', caption: 'Country' }
],
rows: [
{ name: 'CC_STATE', caption: 'State' },
{ name: 'CC_CITY', caption: 'City' }
],
values: [
{ name: 'CC_COMPANY', caption: 'Company' },
{ name: 'CC_EMPLOYEES', caption: 'Employees' },
{ name: 'CCTAXPERCENTAGE', caption: 'Percentage' },
],
filters: []
},
showFieldList: true
};
},
methods: {
},
provide: {
pivotview: [FieldList]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
`
```

When you run the sample, the resulting pivot table will look like this:



	United States			Grand Total		
	Company	Employees	Percentage	Company	Employees	Percenta
CA	16	413279139	0.4800000000...	16	413279139	0.48000
CO	5	1160903623	0.18	5	1160903623	
FL	15	1162362540	0.33	15	1162362540	
GA	20	2358562323	0.12	20	2358562323	
IL	8	707852970	0.1699999999...	8	707852970	0.16999
KS	9	1842349179	0.19	9	1842349179	
LA	6	528468075	0.16	6	528468075	

Explore our Vue Pivot Table sample and ASP.NET Core Web Application to extract data from a Snowflake database and bind to the Pivot Table in this GitHub repository.

Olap in Vue Pivotview component

Getting Started

This section explain steps to create a simple **Pivot Table** with OLAP data source in a Vue environment.

Dependencies

The following list of dependencies are required to use the pivot table component in your application.

`javascript

|-- @syncfusion/ej2-vue-pivotview

|-- @syncfusion/ej2-pivotview

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-data

|-- @syncfusion/ej2-excel-export

|-- @syncfusion/ej2-file-utils

|-- @syncfusion/ej2-compression

|-- @syncfusion/ej2-pdf-export

|-- @syncfusion/ej2-file-utils

|-- @syncfusion/ej2-compression

|-- @syncfusion/ej2-calendars

|-- @syncfusion/ej2-charts

|-- @syncfusion/ej2-svg-base

|-- @syncfusion/ej2-inputs

|-- @syncfusion/ej2-buttons

|-- @syncfusion/ej2-dropdowns

|-- @syncfusion/ej2-lists

|-- @syncfusion/ej2-popups

|-- @syncfusion/ej2-navigations

|-- @syncfusion/ej2-grids

`

Setup Vue Environment

You can use Vue CLI to setup your vue application. To install Vue CLI use the following commands.

`bash

npm install -g @vue/cli

npm install -g @vue/cli-init

`

Create a Vue Application

Create a new Vue application using below Vue CLI command.

`bash

vue init webpack-simple quickstart

`

Now, the application is created in the **quickstart** demo folder. Run the following command one-by-one to navigate to the **quickstart** demo folder, and install the required **npm** dependent packages.

`bash

cd quickstart

npm install

`

Adding Syncfusion pivot table package

All the available Essential JS 2 packages are published in npmjs.com registry. To install pivot table component, use the following command.

`bash

npm install @syncfusion/ej2-vue-pivotview --save

`

The **--save** will instruct NPM to include the pivot table package inside the **dependencies** section of the **package.json**.

Registering pivot table Component

You can register the pivot table component in your application by using the Vue.use() in **App.vue** file as follows.

`ts

import { PivotViewPlugin } from '@syncfusion/ej2-vue-pivotview';

Vue.use(PivotViewPlugin);

`

Registering PivotViewPlugin in vue, will register the pivot table component along with its required child directives globally.

Adding CSS Reference

Import pivot table and its dependent components styles as given below in <style> section of the **App.vue** file.

`

<style>

<!-- Material theme used for this sample -->

@import '../node_modules/@syncfusion/ej2-base/styles/material.css';

@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';

@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';

@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';

@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';

@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';

@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';

Copyright © 2001 -2024 Syncfusion Inc.

@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';

@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';

@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";

</style>

`

Adding pivot table component

Add the EJ2 Vue pivot table using <ejs-pivotview> to the <template> section of the **App.vue** file in src directory. The properties of pivot table can be assigned in <ejs-pivotview> tag and that can be bounded under **data** section.

`

<template>

<ejs-pivotview></ejs-pivotview>

</template>

<script>

import Vue from "vue";

import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";

Vue.use(PivotViewPlugin);

export default {

data () {

return {

}

}

}

</script>

<style>

@import "@syncfusion/ej2-vue-pivotview/styles/material.css";

</style>

`

Next we need to add the the following code in **App.vue** file to populate pivot table with a sample OLAP data source. Refer here to know the more details about OLAP data binding.

`

<template>

```
<div id="app">
<ejs-pivotview :dataSourceSettings="dataSourceSettings"> </ejs-pivotview>
</div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data () {
return {
dataSourceSettings: {
catalog: 'Adventure Works DW 2008 SE',
cube: 'Adventure Works',
providerType: 'SSAS',
enableSorting: true,
url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
localeIdentifier: 1033
}
}
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
`
```

Adding OLAP cube elements to row, column, value and filter axes

Now that pivot table is initialized and assigned with sample OLAP data source, will further move to showcase the component by organizing appropriate OLAP cube elements in rows, columns, values and filters axes.

In dataSourceSettings property, four major axes rows, columns, values and filters plays a vital role in defining and organizing OLAP cube elements from the bound data source, to render the entire pivot table component in a desired format.

Copyright © 2001 -2024 Syncfusion Inc.

rows – Collection of OLAP cube elements (such as Hierarchies, NamedSet, Calculated Members etc.,) that needs to be displayed in row axis of the pivot table.

columns – Collection of OLAP cube elements (such as Hierarchies, NamedSet, Calculated Members etc.,) that needs to be displayed in column axis of the pivot table.

values – Collection of OLAP cube elements (such as Measures, Calculated Measures) that needs to be displayed as aggregated numeric values in the pivot table.

filters - Collection of OLAP cube elements (such as Hierarchies and Calculated Members) that would act as master filter over the data bound in row, column and value axes of the pivot table.

In-order to define each OLAP cube element in the respective axis, the following basic properties should be set.

- name: It allows to set the unique name of the hierarchies, named set, measures, calculated members etc., from the bound OLAP data source. It's casing should match exactly like in the data source and if not set properly, the pivot table will be rendered as empty.
- caption: It allows to set the caption, which is the alias name of the unique name that needs to be displayed in the pivot table. If not provided, unique name will be displayed.

In this sample, "Product Categories" is added in column, "Customer Geography" in row, and "Customer Count" and "Internet Sales Amount" in value axes respectively.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar } from "@syncfusion/ej2-vue-
pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
            catalog: 'Adventure Works DW 2008 SE',
            cube: 'Adventure Works',
            providerType: 'SSAS',
            enableSorting: true,
            url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
            localeIdentifier: 1033,
            rows: [
                { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
            ],
            columns: [
                { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
                { name: '[Measures]', caption: 'Measures' },
            ],
```

```
            values: [
                { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
                { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
            ],
            filters: [
                { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
            ],
            filterSettings: [
                {
                    name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                        '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                    levelCount: 3
                }
            ]
        },
        height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs13" %}

Applying formatting to measures

Formatting defines a way in which values should be displayed in pivot table. For example, format **“C0”** denotes the values should be displayed in currency pattern without decimal points. To do so, define the formatSettings with its name and format properties. In this sample, the name property is set as "[Measures].[Internet Sales Amount]", a measure from value axis and its format is set as "C0". Likewise, we can set format for other measures as well.

Only measures from value axis, which is in the form of numeric data values are applicable for formatting.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar } from "@syncfusion/ej2-vue-
pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
```

```
            catalog: 'Adventure Works DW 2008 SE',
            cube: 'Adventure Works',
            providerType: 'SSAS',
            enableSorting: true,
            url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
            localeIdentifier: 1033,
            rows: [
                { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
            ],
            columns: [
                { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
                { name: '[Measures]', caption: 'Measures' },
            ],
            values: [
                { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
                { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
            ],
            filters: [
                { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
            ],
            filterSettings: [
                {
                    name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                        '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                    levelCount: 3
                }
            ],
            formatSettings: [{ name: '[Measures].[Internet Sales Amount]',
format: 'C0' }]
        },
        height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs14" %}

Enable grouping bar

The Grouping Bar feature automatically populates OLAP cube elements from the bound data source and allows end users to drag OLAP cube elements between different axes such as rows, columns, values and filters, and change pivot view at runtime. Sorting, filtering and removing of elements is also possible. It can be enabled by setting the showGroupingBar property to **true** and by injecting the **GroupingBar** module as follows.

If the **GroupingBar** module is not injected, the grouping bar will not be rendered with the pivot table component.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showGroupingBar="showGroupingBar"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar } from "@syncfusion/ej2-vue-
pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
            catalog: 'Adventure Works DW 2008 SE',
            cube: 'Adventure Works',
            providerType: 'SSAS',
            enableSorting: true,
            url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
            localeIdentifier: 1033,
            rows: [
                { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
            ],
            columns: [
                { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
                { name: '[Measures]', caption: 'Measures' },
            ],
            values: [
                { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
                { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
            ],
            filters: [
                { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
            ],
            filterSettings: [
                {
                    name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                        '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                    levelCount: 3
                }
            ]
      },
      height: 350,
      showGroupingBar: true
    }
  },
  provide: {
        pivotview: [GroupingBar]
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs15" %}

Enable pivot field list

The component provides a built-in Field List similar to Microsoft Excel. It allows you to add or remove OLAP cube elements and also rearrange the OLAP cube elements between different axes, including rows, columns, values and filters along with filter and sort options dynamically at runtime. It can be enabled by setting the showFieldList property to **true** and by injecting the **FieldList** module as follows.

If the **FieldList** module is not injected, the Field List will not be rendered with the pivot table component.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
            catalog: 'Adventure Works DW 2008 SE',
            cube: 'Adventure Works',
            providerType: 'SSAS',
            enableSorting: true,
            url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
            localeIdentifier: 1033,
            rows: [
                { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
            ],
            columns: [
                { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
                { name: '[Measures]', caption: 'Measures' },
            ],
            values: [
                { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
                { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
            ],
            filters: [
                { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
            ],
```

```
            filterSettings: [
                {
                    name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                        '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                    levelCount: 3
                }
            ]
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs16" %}

Exploring filter axis

The filter axis contains collection of OLAP cube elements such as hierarchies and calculated members that would act as master filter over the data bound in rows, columns and values axes of the pivot table. The OLAP cube elements along with filter members could be set to filter axis either through report via code behind or by dragging and dropping OLAP cube elements from other axes to filter axis via grouping bar or field list at runtime.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
            catalog: 'Adventure Works DW 2008 SE',
            cube: 'Adventure Works',
            providerType: 'SSAS',
            enableSorting: true,
            url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
            localeIdentifier: 1033,
            rows: [
                { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
```

```
            ],
            columns: [
                { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
                { name: '[Measures]', caption: 'Measures' },
            ],
            values: [
                { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
                { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
            ],
            filters: [
                { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
            ],
            filterSettings: [
                {
                    name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                        '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                    levelCount: 3
                }
            ]
        },
        height: 350,
        showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs17" %}

Calculated field

The calculated field allows user to insert or add a new calculated field based on the available OLAP cube elements from the bound data source. Calculated fields are nothing but customized dimensions or measures that are newly created based on the user-defined expression.

The two types of calculated fields are as follows:

- **Calculated Measure** – Creates a new measure through user-defined expression.
- **Calculated Dimension** – Creates a new dimension through user-defined expression.

It can be customized using the calculatedFieldsSettings property through code behind. The setting required for calculate field feature at code behind are:

- name: It allows to set the unique name for new calculated field.
- formula: It allows to set the user-defined expression.

Copyright © 2001 -2024 Syncfusion Inc.

- hierarchyUniqueName: It allows to specify dimension unique name whose hierarchies alone should be used in the expression. This will be applicable only for calculated dimension.
- formatString: It allows to set the format string for the resultant calculated field.

You need to set isCalculatedField property to true, while adding calculated fields to respective axis through code behind.

Also calculated fields can be added at run time through the built-in dialog. The dialog can be enabled by setting the allowCalculatedField property to **true** and by injecting the **CalculatedField** module as follows. You will see a button enabled in the Field List UI automatically to invoke the calculated field dialog and perform necessary operation.

If the **CalculatedField** module is not injected, the calculated field dialog will not be rendered with the pivot table component. Moreover calculated measure can be added only in value axis.

APP.VUE

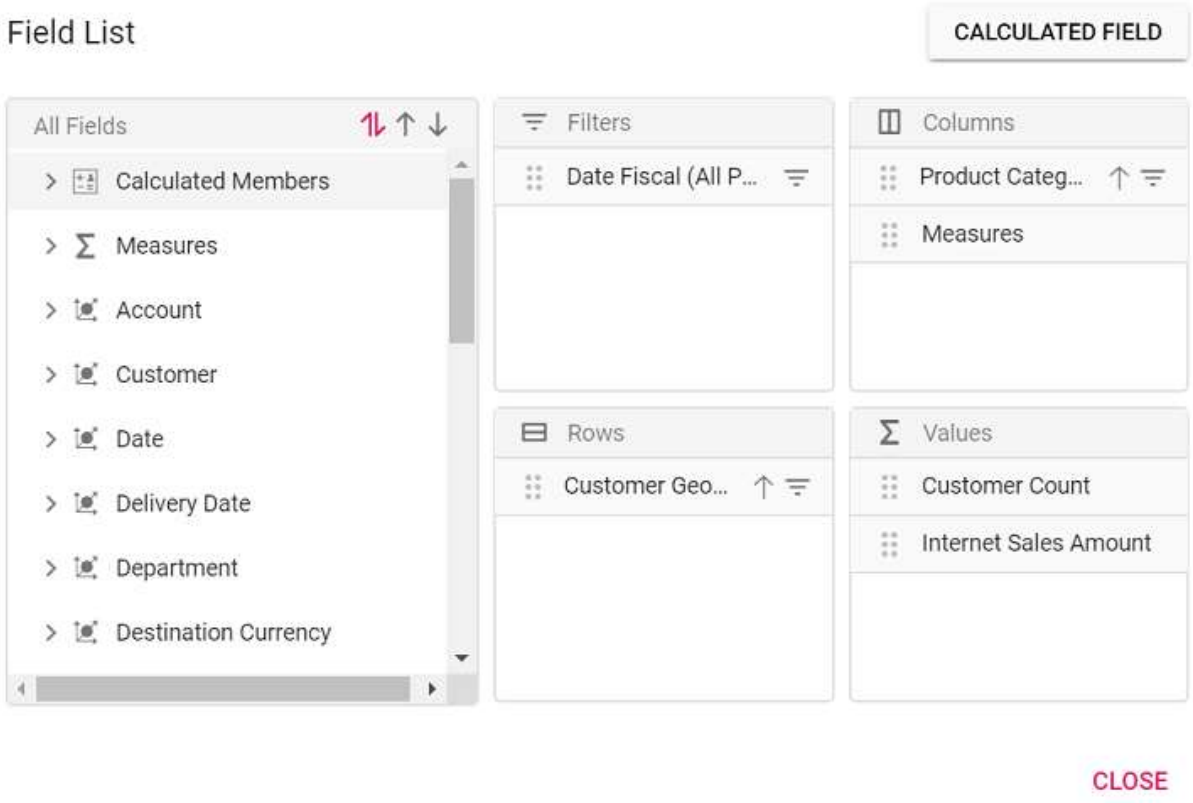
```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
            catalog: 'Adventure Works DW 2008 SE',
            cube: 'Adventure Works',
            providerType: 'SSAS',
            enableSorting: true,
            url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
            localeIdentifier: 1033,
            rows: [
                { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
            ],
            columns: [
                { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
                { name: '[Measures]', caption: 'Measures' },
            ],
            values: [
                { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
                { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' },
                { name: 'Order on Discount', isCalculatedField: true }
            ],
            filters: [
                { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
            ],
```

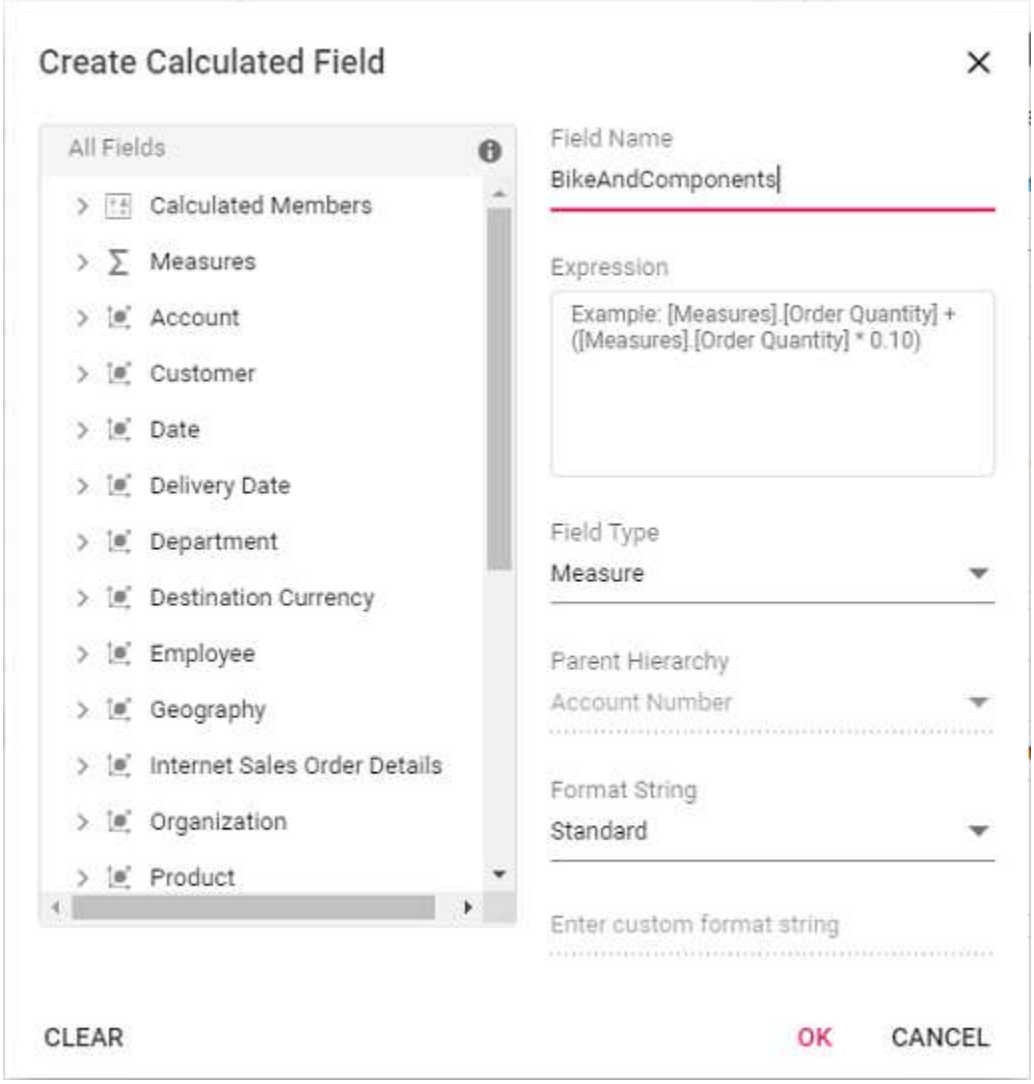
```
            calculatedFieldSettings: [
                {
                    name: 'BikeAndComponents',
                    formula: '([Product].[Product
Categories].[Category].[Bikes] + [Product].[Product
Categories].[Category].[Components] )',
                    hierarchyUniqueName: '[Product].[Product Categories]',
                    formatString: 'Standard'
                },
                {
                    name: 'Order on Discount',
                    formula: '[Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)',
                    formatString: 'Currency'
                }
            ],
            filterSettings: [
                {
                    name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                        '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                    levelCount: 3
                }
            ]
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs18" %}

Users can add a calculated field at runtime through the built-in dialog by using the following steps.

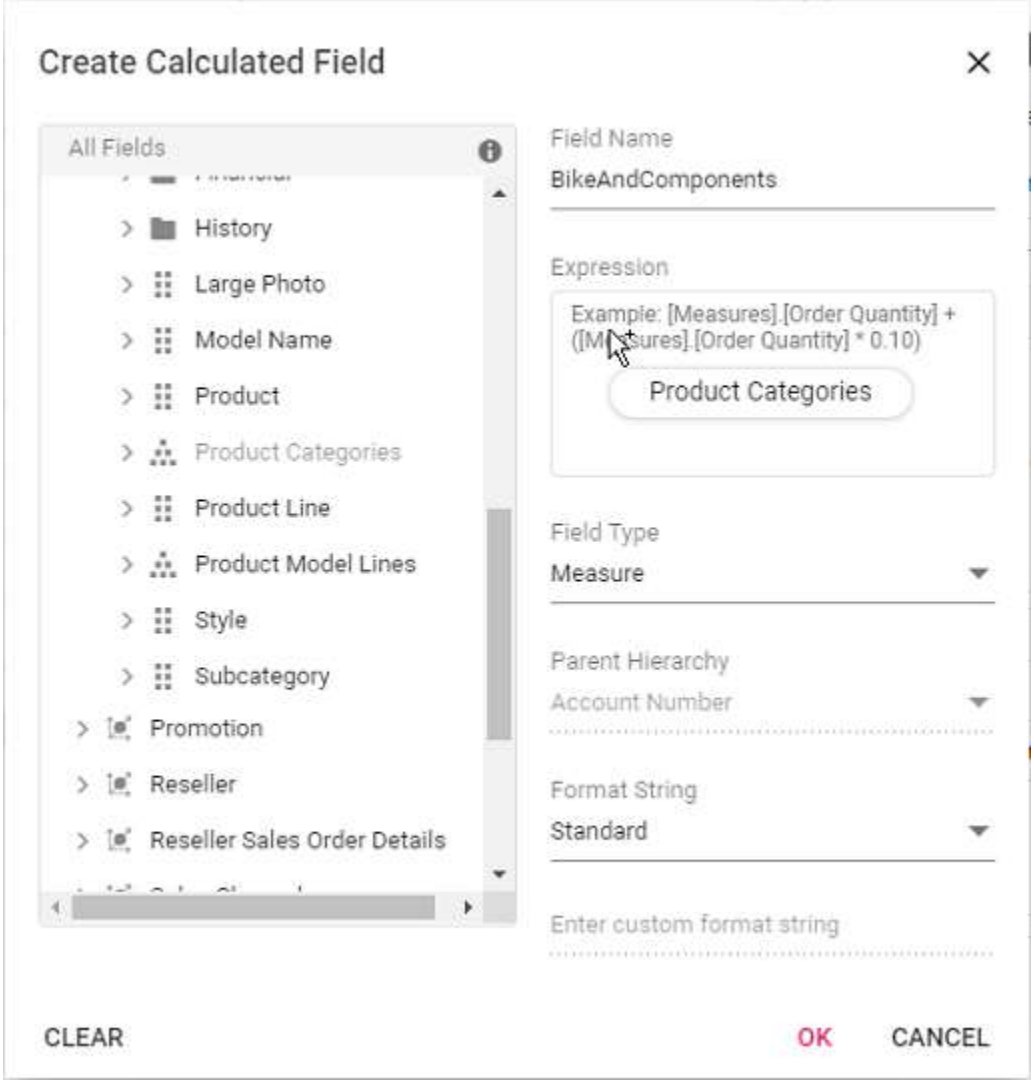
Step 1: Click the "CALCULATED FIELD" button in the field list dialog positioned at the top right corner. The calculated field dialog will be opened now. Enter the name of the calculated field to be created.


Field List
CALCULATED FIELD
All Fields
Calculated Members
Measures
Account
Customer
Date
Delivery Date
Department
Destination Currency
Filters
Date Fiscal (All P...
Columns
Product Categ...
Measures
Rows
Customer Geo...
Values
Customer Count
Internet Sales Amount
CLOSE

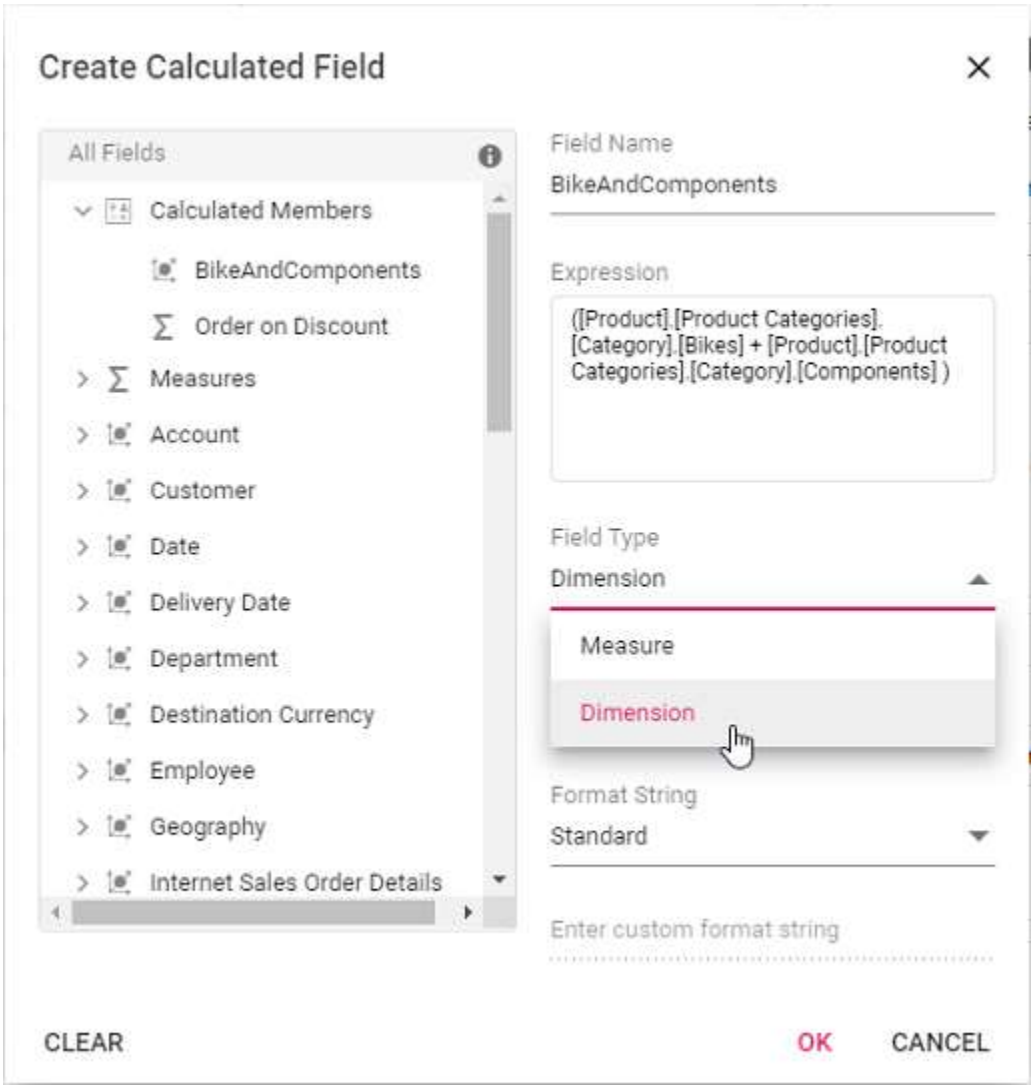


Step 2: Frame the expression by dragging and dropping the fields from the tree view on the left side of the dialog using simple arithmetic operators. **Example:** "IIF([Measures].[Internet Sales Amount]^0.5 > 100, [Measures].[Internet Sales Amount]*100, [Measures].[Internet Sales Amount]/100)". Please refer here to learn more about the supported operators and functions to frame the expression.

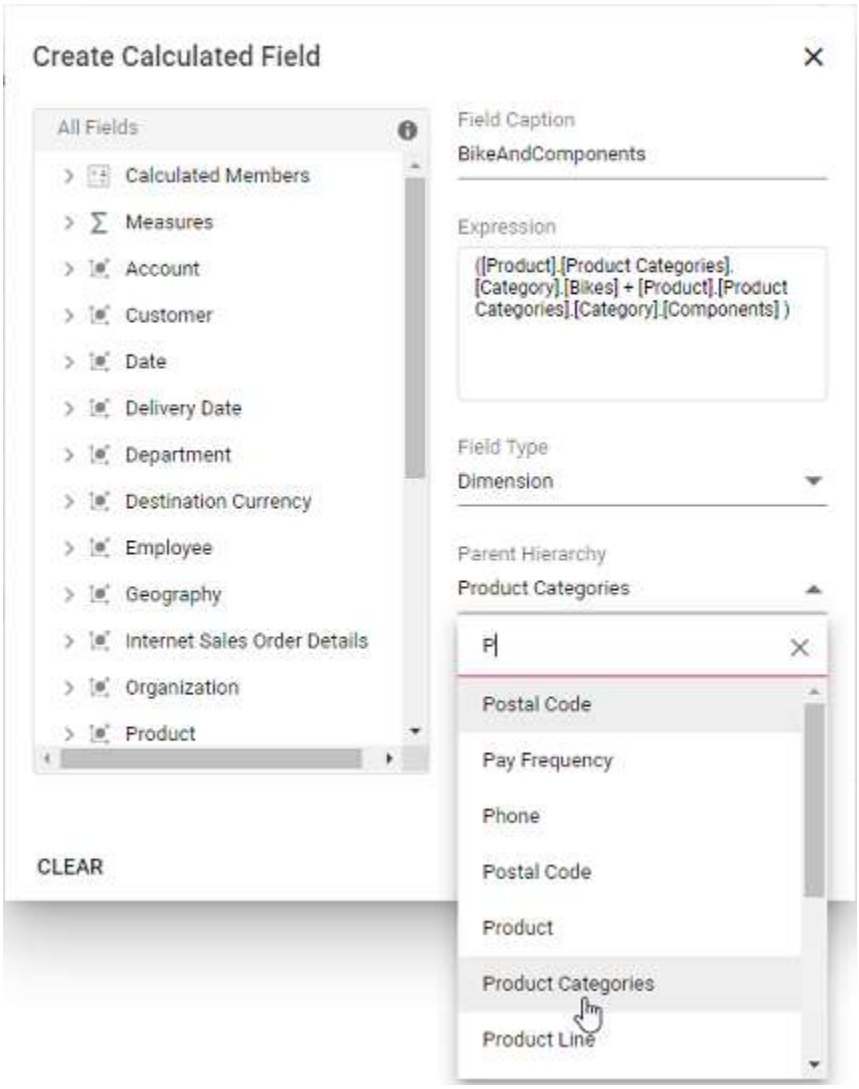
Copyright © 2001 -2024 Syncfusion Inc.



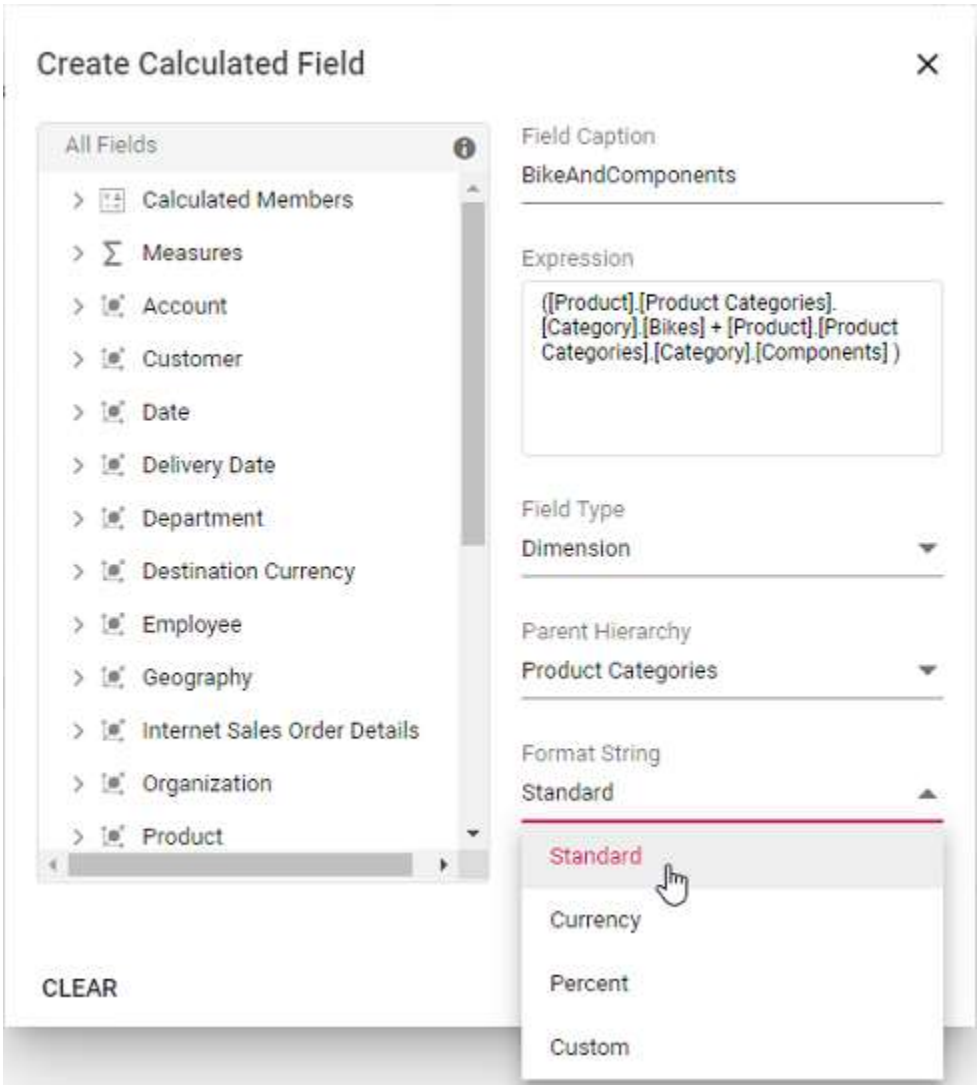
Step 3: Confirm the type of the field to be created - calculated measure or calculated dimension.



Step 4: Choose the parent hierarchy of the calculated field. NOTE: It is only applicable to the calculated dimension.



Step 5: Then select the format string from the drop-down list and finally click "OK".



	Accessories		
	Customer Count	Internet Sales Amount	Order on Discount
Australia	2,905	$138,690.63	$68,124.10
Canada	1,230	$103,377.85	$68,124.10
France	1,505	$63,406.78	$68,124.10
Germany	1,535	$62,232.59	$68,124.10

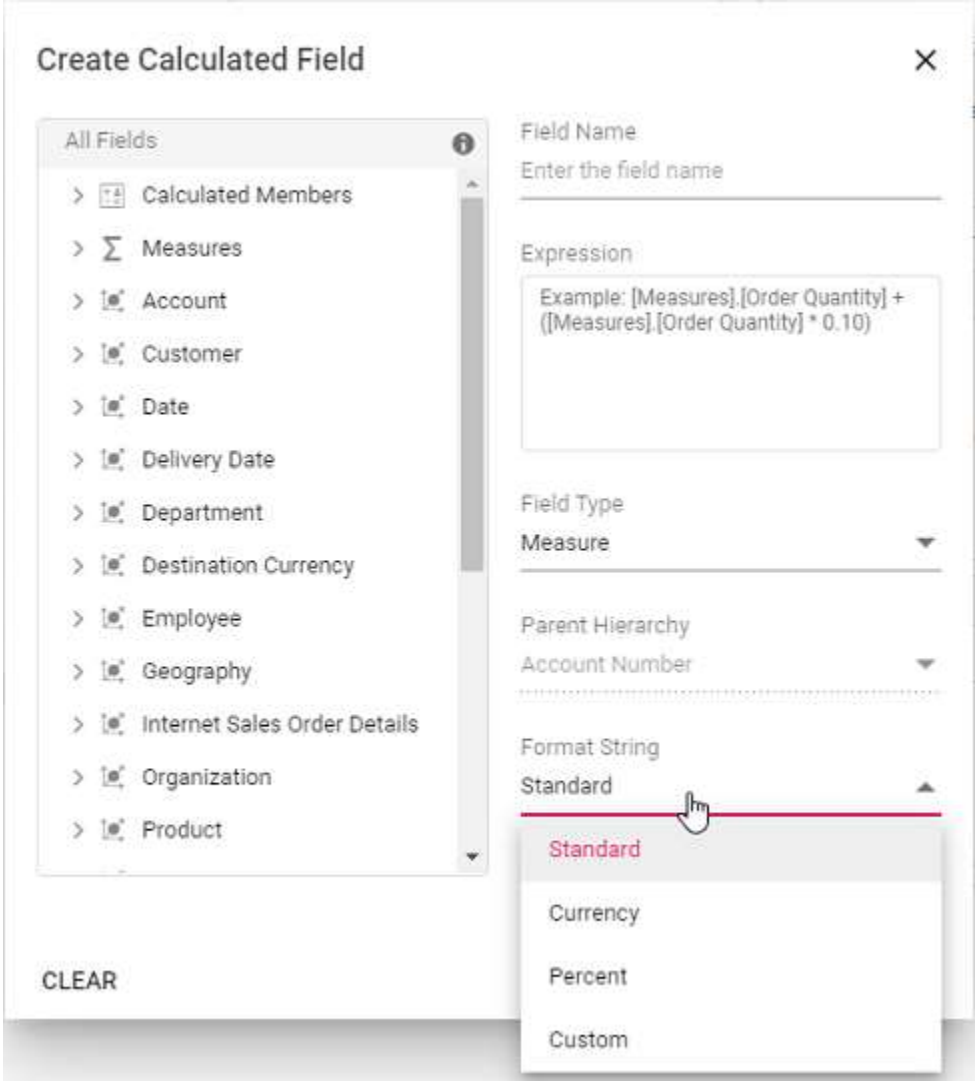
Copyright © 2001 -2024 Syncfusion Inc.

Format String

Allows you to specify the required format string while creating new calculated field. Supported format strings are:

- **Standard** - Denotes the numeric type.
- **Currency** - Denotes the currency type.
- **Percent** - Denotes the percentage type.
- **Custom** - Denotes the custom format. For example: "###0.##0#". This shows the value "9584.3" as "9584.300."

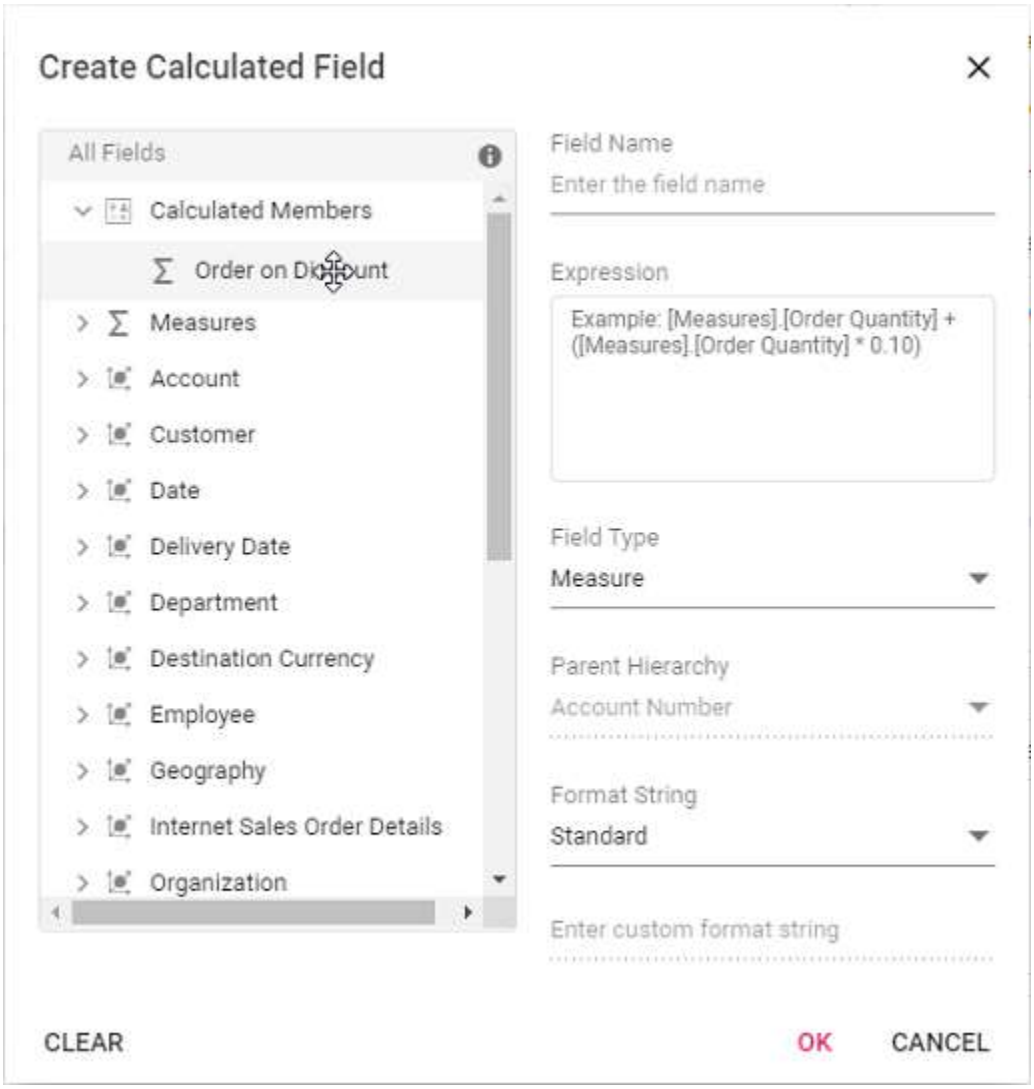
By default, **Standard** will be selected from the drop down list.

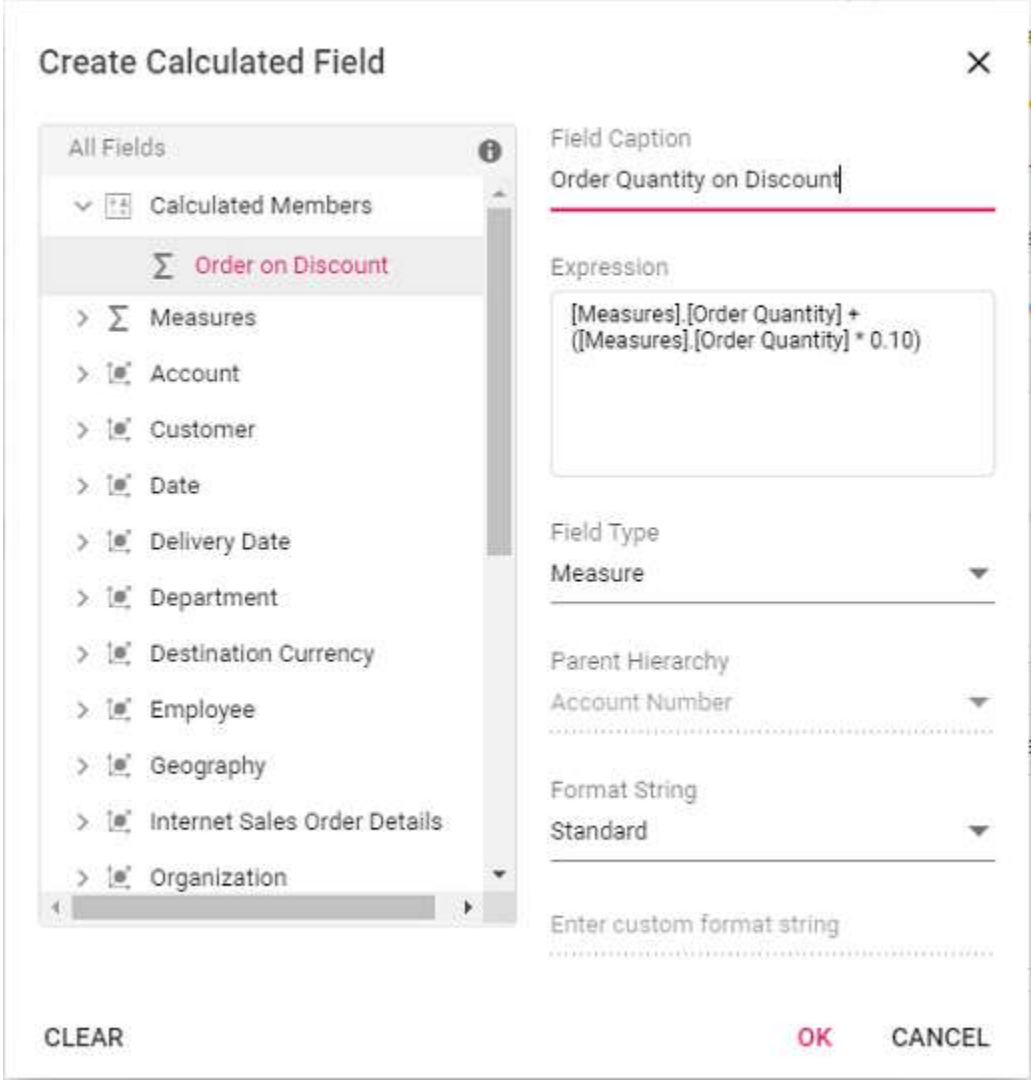


Renaming the existing calculated field

Existing calculated field can be renamed only through the UI at runtime. To do so, open the calculated field dialog, click the target field. User can now see the existing name getting displayed in the text box at the top of the dialog. Now, change the name based on user requirement and click "OK".

<!-- markdownlint-disable MD012 -->

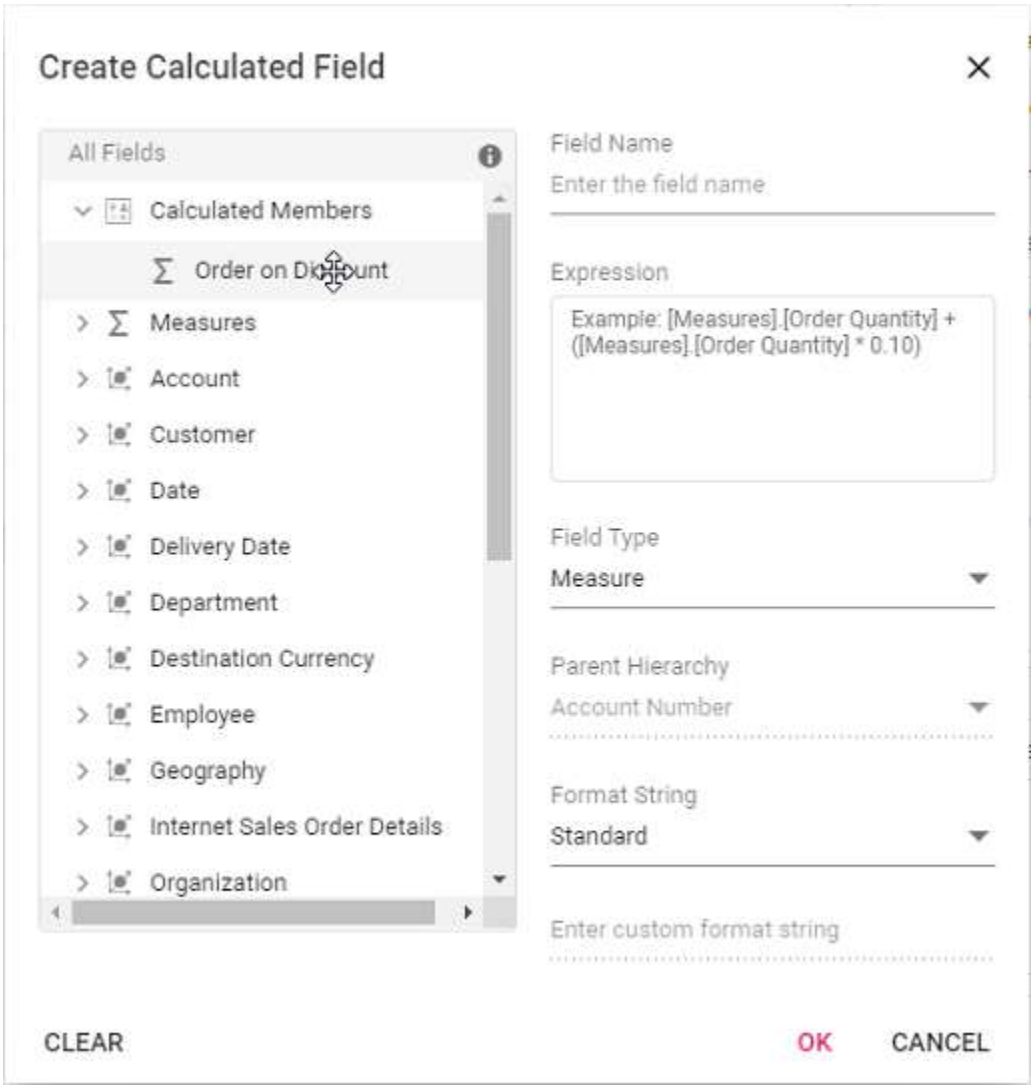

Create Calculated Field
All Fields
Calculated Members
Order on Discount
Measures
Account
Customer
Date
Delivery Date
Department
Destination Currency
Employee
Geography
Internet Sales Order Details
Organization
Field Name
Enter the field name
Expression
Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)
Field Type
Measure
Parent Hierarchy
Account Number
Format String
Standard
Enter custom format string
CLEAR
OK
CANCEL

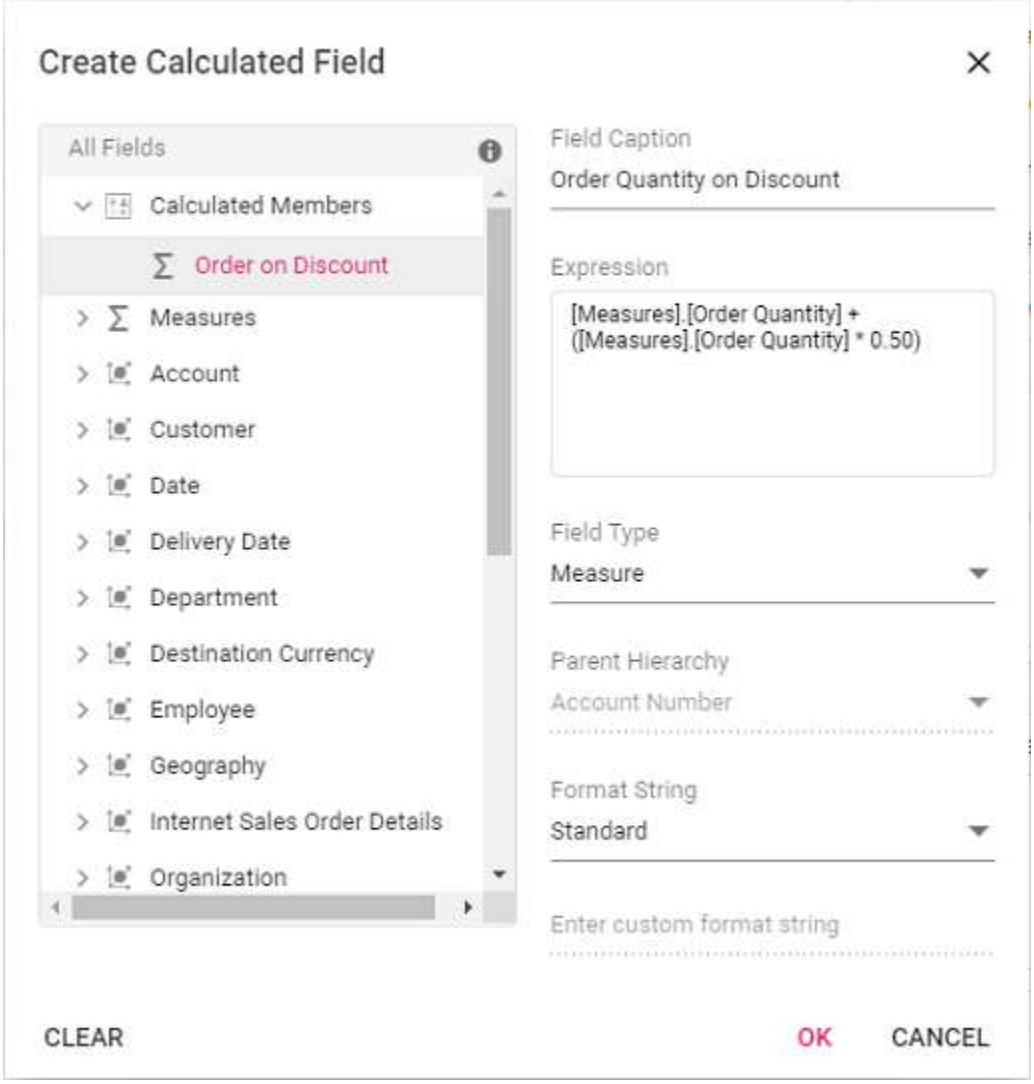


Editing the existing calculated field formula

Existing calculated field formula can be edited only through the UI at runtime. To do so, open the calculated field dialog, click the target field. User can now see the existing expression getting displayed in a "Expression" section. Now, change the expression based on user requirement and click "OK".

Copyright © 2001 -2024 Syncfusion Inc.

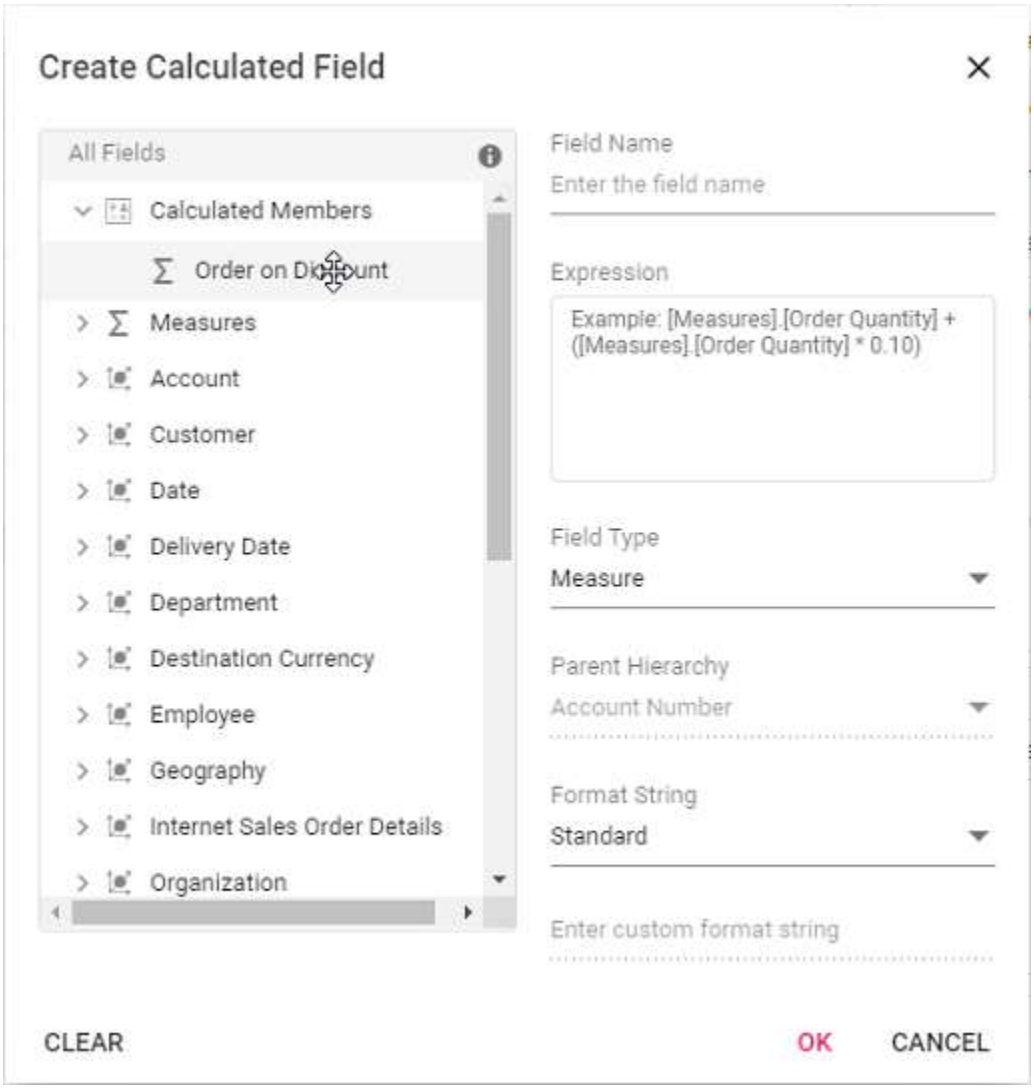

Create Calculated Field
All Fields
Calculated Members
Order on Discount
Measures
Account
Customer
Date
Delivery Date
Department
Destination Currency
Employee
Geography
Internet Sales Order Details
Organization
Field Name
Enter the field name
Expression
Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)
Field Type
Measure
Parent Hierarchy
Account Number
Format String
Standard
Enter custom format string
CLEAR
OK
CANCEL

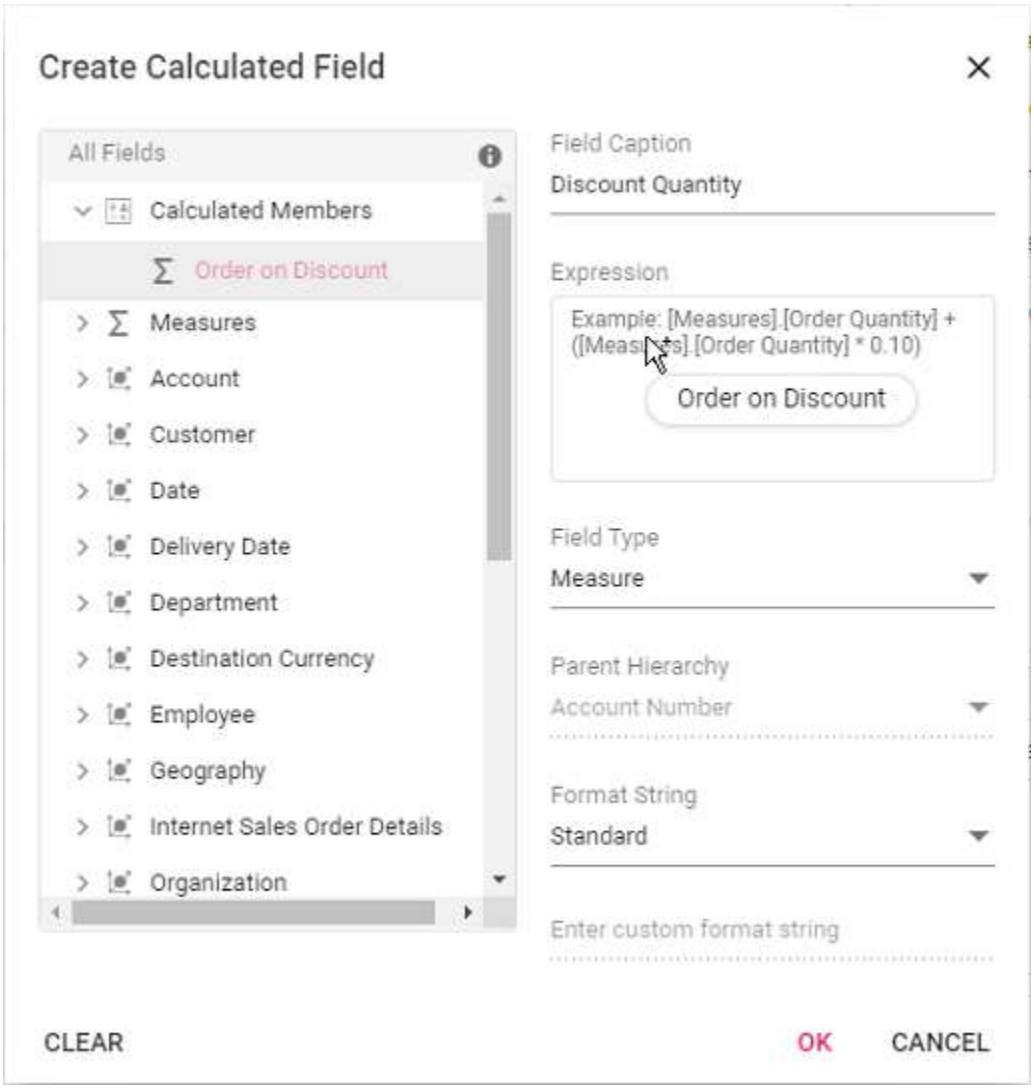


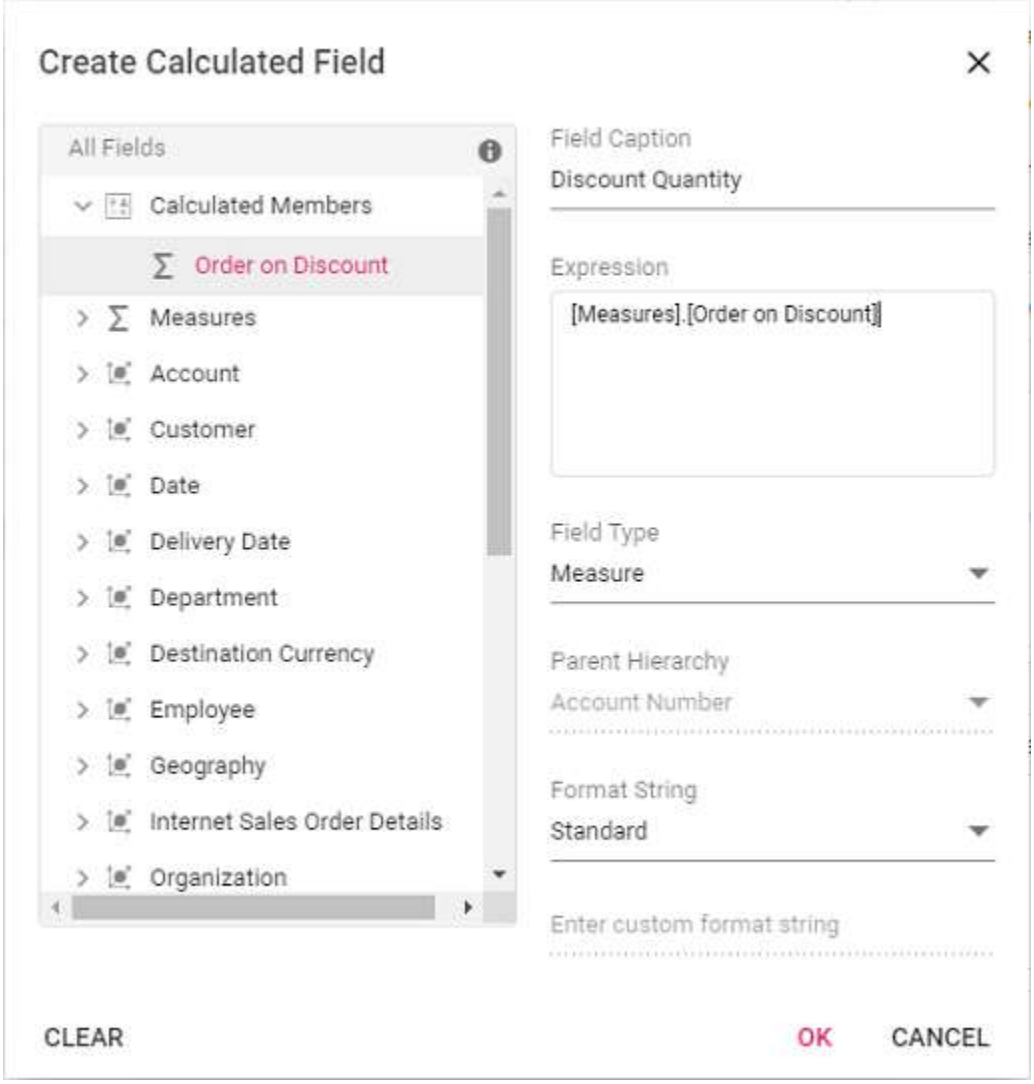
Reusing the existing formula in a new calculate field

While creating a new calculated field, if user wants to the add the formula of an existing calculated field, it can be done easily. To do so, simply drag-and-drop the existing calculated field to the "Expression" section.

Copyright © 2001 -2024 Syncfusion Inc.


Create Calculated Field
All Fields
Calculated Members
Order on Discount
Measures
Account
Customer
Date
Delivery Date
Department
Destination Currency
Employee
Geography
Internet Sales Order Details
Organization
Field Name
Enter the field name
Expression
Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)
Field Type
Measure
Parent Hierarchy
Account Number
Format String
Standard
Enter custom format string
CLEAR
OK
CANCEL

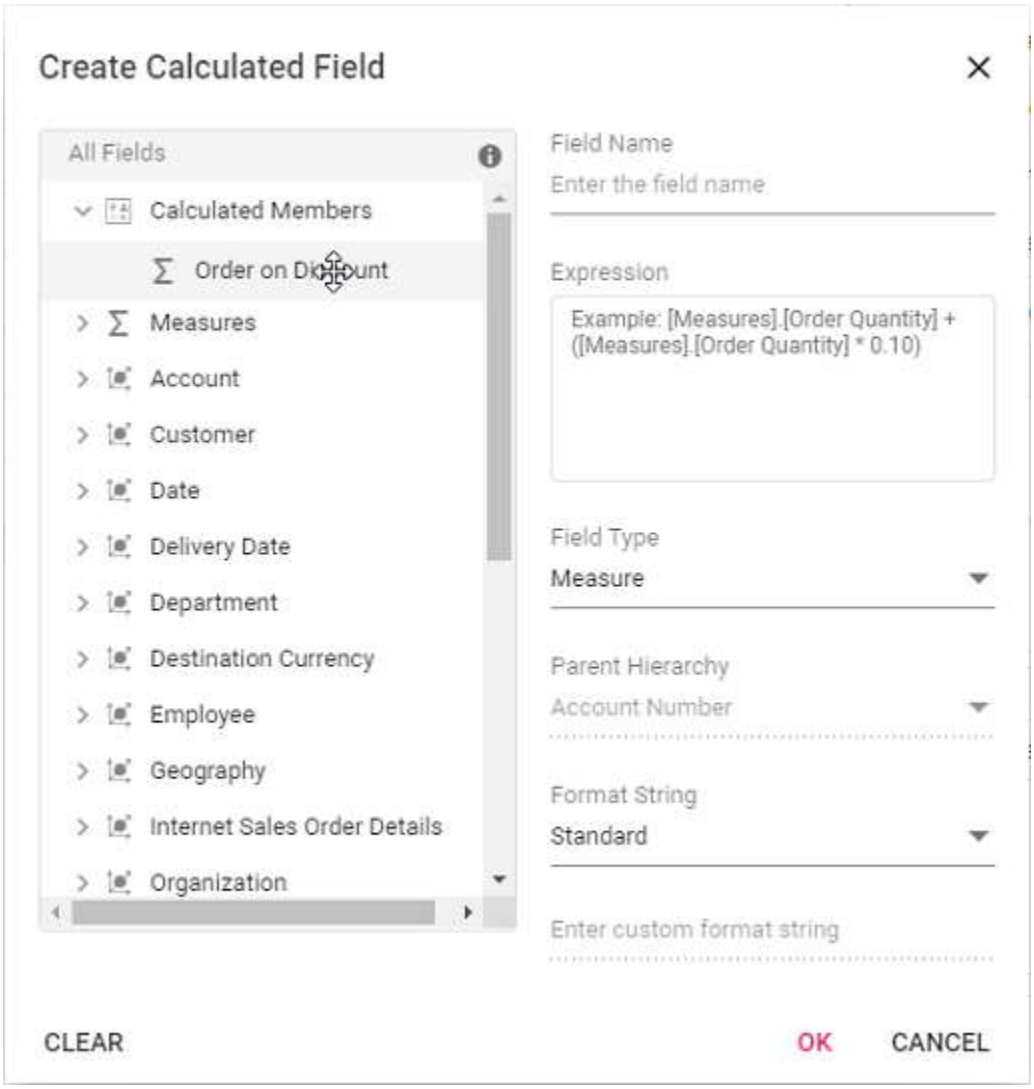

Create Calculated Field
All Fields
Calculated Members
Order on Discount
Measures
Account
Customer
Date
Delivery Date
Department
Destination Currency
Employee
Geography
Internet Sales Order Details
Organization
Field Caption
Discount Quantity
Expression
Example: [Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)
Order on Discount
Field Type
Measure
Parent Hierarchy
Account Number
Format String
Standard
Enter custom format string
CLEAR
OK
CANCEL

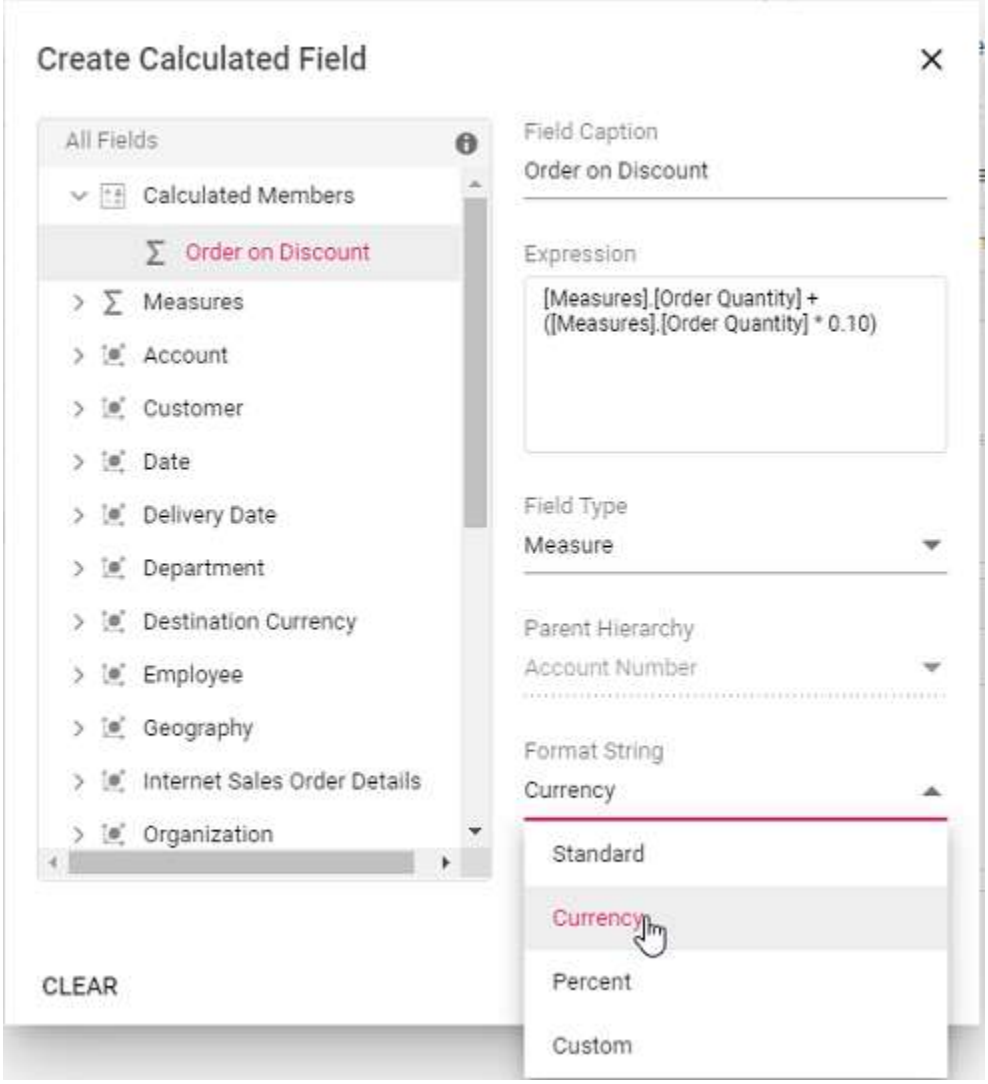


Modifying the existing format string

Existing calculated field's format string can be modified only through the UI at runtime. To do so, open the calculated field dialog and click the target calculated field. User can now see the format string for the existing calculated field getting displayed in a drop-down list. Change the format string based on the requirement and finally click "OK".

Copyright © 2001 -2024 Syncfusion Inc.

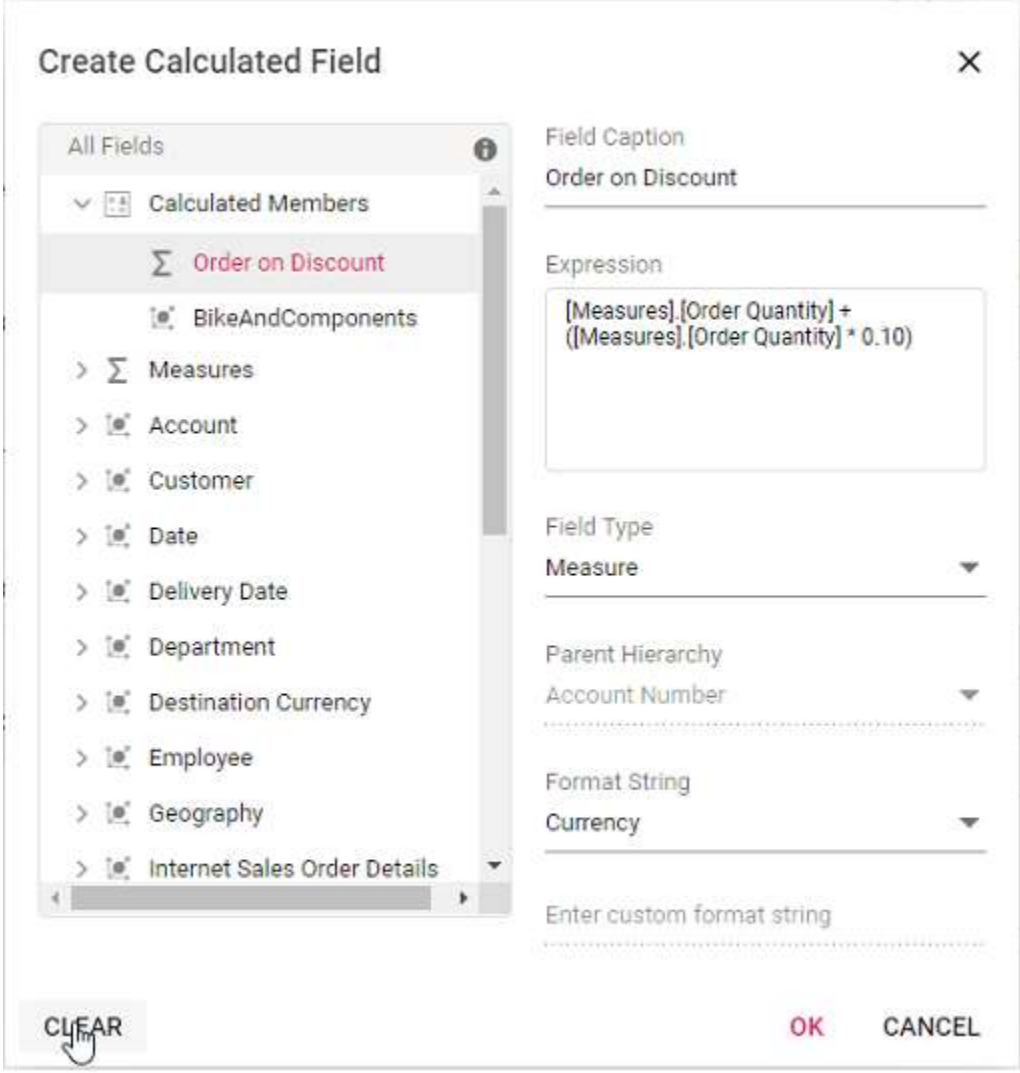

Create Calculated Field
All Fields
Calculated Members
Order on Discount
Measures
Account
Customer
Date
Delivery Date
Department
Destination Currency
Employee
Geography
Internet Sales Order Details
Organization
Field Name
Enter the field name
Expression
Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)
Field Type
Measure
Parent Hierarchy
Account Number
Format String
Standard
Enter custom format string
CLEAR
OK
CANCEL



Clearing the changes while editing the calculated field

Previous changes can be cleared by using the "Clear" option while performing operations such as creating and editing the calculated field. To do so, click the "Clear" button in the bottom left corner of the dialog.

Copyright © 2001 -2024 Syncfusion Inc.



Virtual Scrolling

Allows large amounts of data to be loaded without any performance degradation by rendering rows and columns in relation to the current viewport. Rest of the data will be brought into the viewport dynamically based on vertical or horizontal scroll position. This feature can be enabled by setting the enableVirtualization property to **true**.

To use the virtual scrolling feature, inject the VirtualScroll module into the pivot table.

APP.VUE

```
<template>
  <div id="app">
    <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height">
    </ejs-pivotview>
  </div>
</template>
<script>
import Vue from "vue";
```

```
import { PivotViewPlugin, VirtualScroll } from "@syncfusion/ej2-vue-
pivotview";
Vue.use(PivotViewPlugin);
export default {
  data() {
    return {
      dataSourceSettings: {
        catalog: "Adventure Works DW 2008 SE",
        cube: "Adventure Works",
        providerType: "SSAS",
        enableSorting: true,
        url: "https://bi.syncfusion.com/olap/msmdpump.dll",
        localeIdentifier: 1033,
        rows: [{ name: "[Customer].[Customer]", caption: "Customer" }],
        columns: [
          {
            name: "[Product].[Product Categories]",
            caption: "Product Categories",
          },
          { name: "[Measures]", caption: "Measures" },
        ],
        values: [
          { name: "[Measures].[Customer Count]", caption: "Customer Count"
},
          {
            name: "[Measures].[Internet Sales Amount]",
            caption: "Internet Sales Amount",
          },
        ],
        filters: [{ name: "[Date].[Fiscal]", caption: "Date Fiscal" }],
        filterSettings: [
          {
            name: "[Date].[Fiscal]",
            items: [
              "[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]",
              "[Date].[Fiscal].[Fiscal Year].&[2005]",
            ],
            levelCount: 3,
          },
        ],
        formatSettings: [
          { name: "[Measures].[Internet Sales Amount]", format: "C0" },
        ],
      },
      height: 350,
      enableVirtualization: true,
    };
  },
  provide: {
    pivotview: [VirtualScroll],
  },
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs19" %}

Limitations for virtual scrolling

- The columnWidth property in gridSettings should be in pixels. The percentage value is not accepted.
- Resizing columns and setting the width of individual columns will affect scrolling and is therefore not recommended.
- The grand totals option is not supported by virtual scrolling.

Run the application

The quickstart project is configured to compile and run the application in the browser. Use the following command to run the application.

`bash

npm run dev

`

Output will be displayed as follows.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
            catalog: 'Adventure Works DW 2008 SE',
            cube: 'Adventure Works',
            providerType: 'SSAS',
            enableSorting: true,
            url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
            localeIdentifier: 1033,
            rows: [
                { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
            ],
            columns: [
                { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
                { name: '[Measures]', caption: 'Measures' },
            ],
```

```
            values: [
                { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
                { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' },
                { name: 'Order on Discount', isCalculatedField: true }
            ],
            filters: [
                { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
            ],
            calculatedFieldSettings: [
                {
                    name: 'BikeAndComponents',
                    formula: '([Product].[Product
Categories].[Category].[Bikes] + [Product].[Product
Categories].[Category].[Components] )',
                    hierarchyUniqueName: '[Product].[Product Categories]',
                    formatString: 'Standard'
                },
                {
                    name: 'Order on Discount',
                    formula: '[Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)',
                    formatString: 'Currency'
                }
            ],
            filterSettings: [
                {
                    name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                        '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                    levelCount: 3
                }
            ]
        },
        showFieldList: true,
        height: 350
    }
  },
  provide: {
    pivotview: [FieldList]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs20" %}

Data Binding

To bind OLAP datasource to the pivot table, you need to specify following properties under dataSourceSettings option.

| Properties|Description |

|-----|-----|

| cube| Points the respective cube name from OLAP database.|

| providerType| Points the provider type for pivot table to identify the type of data source.|

| url| Contains the cube URL for establishing the connection (online).|

| catalog| Contains the database name (catalog name) to fetch the data.|

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
            catalog: 'Adventure Works DW 2008 SE',
            cube: 'Adventure Works',
            providerType: 'SSAS',
            enableSorting: true,
            url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
            localeIdentifier: 1033,
            rows: [
                { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
            ],
            columns: [
                { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
                { name: '[Measures]', caption: 'Measures' },
            ],
            values: [
                { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
                { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
            ],
            filters: [
                { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
            ],
            filterSettings: [
                {
                    name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                        '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                    levelCount: 3
                }
            ]
```

```
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs21" %}

Fields

Measures in row axis

By default, the measures are plotted in column axis. To plot those measures in row axis, place the **Measures** button in the row axis as follows.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
            catalog: 'Adventure Works DW 2008 SE',
            cube: 'Adventure Works',
            providerType: 'SSAS',
            enableSorting: true,
            url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
            localeIdentifier: 1033,
            rows: [
                { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
                { name: '[Measures]', caption: 'Measures' },
            ],
            columns: [
                { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
            ],
            values: [
                { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
```

```
                { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
            ],
            filters: [
                { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
            ],
            filterSettings: [
                {
                    name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                        '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                    levelCount: 3
                }
            ]
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs22" %}

Measures in different position

You can place measures in different position in row or column axis either thorough code behind or UI. In this sample, **measures** placed before the dimension in the column axis.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
            catalog: 'Adventure Works DW 2008 SE',
            cube: 'Adventure Works',
            providerType: 'SSAS',
            enableSorting: true,
            url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
            localeIdentifier: 1033,
```

```
            rows: [
                { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
            ],
            columns: [
                { name: '[Measures]', caption: 'Measures' },
                { name: '[Product].[Product Categories]', caption: 'Product
Categories' },
            ],
            values: [
                { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
                { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
            ],
            filters: [
                { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
            ],
            filterSettings: [
                {
                    name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                        '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                    levelCount: 3
                }
            ]
        },
        height: 350,
        showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs23" %}

Named set

Named set is a multidimensional expression (MDX) that returns a set of dimension members, which can be created by combining the cube data, arithmetic operators, numbers, and functions.

You can bind the named sets in the pivot table by setting it's unique name in the name property either in row or column axis and isNamedSet boolean property to **true**. In this sample, we have added "Core Product Group" named set in the column axis.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showFieldList="showFieldList"> </ejs-pivotview>
```

```
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
            catalog: 'Adventure Works DW 2008 SE',
            cube: 'Adventure Works',
            providerType: 'SSAS',
            enableSorting: true,
            url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
            localeIdentifier: 1033,
            rows: [
                { name: '[Customer].[Customer Geography]', caption:
'Customer Geography' },
            ],
            columns: [
                {
                    name: "[Core Product Group]",
                    isNamedSet: true
                },
                { name: '[Measures]', caption: 'Measures' },
            ],
            values: [
                { name: '[Measures].[Customer Count]', caption: 'Customer
Count' },
                { name: '[Measures].[Internet Sales Amount]', caption:
'Internet Sales Amount' }
            ],
            filters: [
                { name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
            ],
            filterSettings: [
                {
                    name: '[Date].[Fiscal]', items:
['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
                        '[Date].[Fiscal].[Fiscal Year].&[2005]'],
                    levelCount: 3
                }
            ]
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs24" %}

Configuring authentication

Users can configure basic authentication information to access the OLAP cube using the authentication property. The settings required to configure are as follows:

- userName: It allows the user to set a username that recognizes the basic authentication of the IIS.
- password: It allows to set the appropriate password.

If the user does not configure the authentication, a default popup will appear in the browser to get the authentication information.

`

```
<template>
<div id="app">
<ejs-pivotview :dataSourceSettings="dataSourceSettings" :height= "height"
:showFieldList="showFieldList"> </ejs-pivotview>
</div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data () {
return {
dataSourceSettings: {
catalog: 'Adventure Works DW 2008 SE',
cube: 'Adventure Works',
providerType: 'SSAS',
enableSorting: true,
url: 'https://bi.syncfusion.com/olap/msmdpump.dll',
localeIdentifier: 1033,
rows: [
{ name: '[Customer].[Customer Geography]', caption: 'Customer Geography' },
],
```

```
columns: [
{ name: '[Product].[Product Categories]', caption: 'Product Categories' },
{ name: '[Measures]', caption: 'Measures' },
],
values: [
{ name: '[Measures].[Customer Count]', caption: 'Customer Count' },
{ name: '[Measures].[Internet Sales Amount]', caption: 'Internet Sales Amount' }
],
filters: [
{ name: '[Date].[Fiscal]', caption: 'Date Fiscal' },
],
filterSettings: [
{
name: '[Date].[Fiscal]', items: ['[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]',
'[Date].[Fiscal].[Fiscal Year].&[2005]'],
levelCount: 3
}
],
authentication: {
userName: 'username',
password: 'password'
}
},
height: 350,
showFieldList: true
}
},
provide: {
pivotview: [FieldList]
}
}
</script>
<style>
```

@import "@syncfusion/ej2-vue-pivotview/styles/material.css";

</style>

`

Roles

SQL Server Analysis Services uses roles to limit data access within a cube. Each role defines a set of permissions that can be granted to a single user or groups of users. It is used to manage security by limiting access to sensitive data and determining who has access to and can change the cube. It can be configured using the roles property in dataSourceSettings.

The roles property can be used to specify one or more roles to the OLAP cube, separated by commas.

`

<template>

<ejs-pivotview :dataSourceSettings="dataSourceSettings" :height= "height"> </ejs-pivotview>

</template>

<script>

import Vue from "vue";

import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";

Vue.use(PivotViewPlugin);

export default {

data () {

return {

dataSourceSettings: {

catalog: 'Adventure Works DW 2008 SE',

cube: 'Adventure Works',

providerType: 'SSAS',

roles: 'Role1',

url: 'https://bi.syncfusion.com/olap/msmdpump.dll',

localeIdentifier: 1033,

rows: [

{ name: '[Customer].[Customer Geography]', caption: 'Customer Geography' },

],

columns: [

{ name: '[Product].[Product Categories]', caption: 'Product Categories' },

```
{ name: '[Measures]', caption: 'Measures' },
],
values: [
{ name: '[Measures].[Customer Count]', caption: 'Customer Count' },
{ name: '[Measures].[Internet Sales Amount]', caption: 'Internet Sales Amount' }
]
},
height: 350
}
},
provide: {
pivotview: []
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
`
```

OLAP Cube: Elements

Field list

The field list, aka cube dimension browser, is a tree view like structure that organizes the cube elements such as dimensions, hierarchies, measures, etc., from the selected cube into independent logical groups.

Types of node in field list

- **Display folder**: A folder that contains a set of similar elements.
- **Measure**: Quantity available for analysis.
- **Dimension**: A name given to the parts of the cube that categorizes data.
- **Attribute Hierarchy**: Level of attributes down the hierarchy.
- **User-defined Hierarchy**: Members of a dimension in a hierarchical structure.
- **Level**: Denotes a specific level in the category.
- **Named Set**: A collection of tuples and members, that can be defined and saved as a part of cube definition for later use.

Measure

In a cube, a measure is a set of values that are based on a column in the cube's fact table and are usually numeric. The measures are the central values of a cube that are analyzed. That is, measures are the

numeric data of primary interest to users browsing a cube. You can select measures depend on the types of users request. Some common measures are sales, costs, expenditures, and production count.

Dimension

A simple dimension object is composed of basic information such as name, hierarchy, level, and members. You can create a dimension element by specifying its name and providing the hierarchy and level name. The dimension element contains the hierarchical details and information about each included level elements in that hierarchy. A hierarchy can have any number of level elements and the level elements can have any number of members and the member elements can have any number of child members.

Hierarchy

Each element of a dimension can be summarized using a hierarchy. The hierarchy is a series of parent-child relationship, where a parent member represents the consolidation of members which are its children. Parent members can be further aggregated as the children of another parent. For example, May 2005 can be summarized into Second Quarter 2005 which in turn would be summarized in the year 2005.

Level

Level element is the child of hierarchy element which contains a set of members, each of which has the same rank within a hierarchy.

Attribute hierarchy

Attribute hierarchy contains the following levels:

- A leaf level contains distinct attribute member, and each member of the leaf level is known as a leaf member.
- Intermediate levels if the attribute hierarchy is a parent-child hierarchy.
- An optional (all) level contains the aggregated value of the attribute hierarchy's leaf members, with the member of the (all) level also known as the (all) member.

User-defined hierarchy

User-defined hierarchy organizes the members of a dimension into hierarchical structure and provides navigation paths in a cube. For example, take a dimension table that supports three attributes such as year, quarter, and month. The year, quarter, and month attributes are used to construct a user-defined hierarchy, named Calendar, in the time dimension that relates to all levels.

Differentiating user-defined hierarchy and attribute hierarchy

- User-defined hierarchy contains more than one level whereas attribute hierarchy contains only one level.
- User-defined hierarchy provides the navigation path between the levels taken from attribute hierarchies of the same dimension.
- The attribute hierarchy and the user-defined hierarchy are represented in different ways as shown in the following table.

Named set

A named set is a collection of tuples and members, which can be defined and saved as a part of the cube definition. Named set records reside inside the sets folder, which is under a dimension element. These elements can be dragged to rows or columns axis via grouping bar or field list at runtime. To work with a

Copyright © 2001 -2024 Syncfusion Inc.

lengthy, complex, or commonly used expression easier, Multidimensional Expressions (MDX) allows you to define a named set.

Calculated field

The calculated field allows user to insert or add a new calculated field based on the available OLAP cube elements from the bound data source. Calculated fields are nothing but customized dimensions or measures that are newly created based on the user-defined expression.

The two types of calculated fields are as follows:

- **Calculated Measure** – Creates a new measure through user-defined expression.
- **Calculated Dimension** – Creates a new dimension through user-defined expression.

Symbolic representation of the nodes inside field list

| Icon|Name | Node type | Is Draggable |

|-----|-----|-----------|--------------|

| | Display Folder| Display Folder| False|

| |Measure| Measure| False|

| | Dimension| Dimension| False|

| | User Defined Hierarchy| Hierarchy| True|

| | Attribute Hierarchy| Hierarchy| True|

|

  | Levels (in order)| Level Element| True|

| | Named Set| Named Set| True|

In general, the Pivot Table is created using the built-in engine for given data source. This is an optional feature that allows you to create the Pivot Table with a server-side pivot engine and external data binding. And this option is applicable only for relational data source.

Server side pivot engine in Vue Pivotview component

This section briefs the Syncfusion assembly Syncfusion.Pivot.Engine, which is used in a server-side application to perform all Pivot calculations such as aggregation, filtering, sorting, grouping, and so on, and only the information to be displayed in the Pivot Table's viewport is passed to the client-side (browser) via web service (Web API) rather than the entire data source. It reduces network traffic and improves the rendering performance of the Pivot Table, especially when dealing with large amounts of data. It also works best with virtual scrolling enabled and supports all the Pivot Table's existing features.

Quick steps to render the Pivot Table by using the server-side Pivot Engine

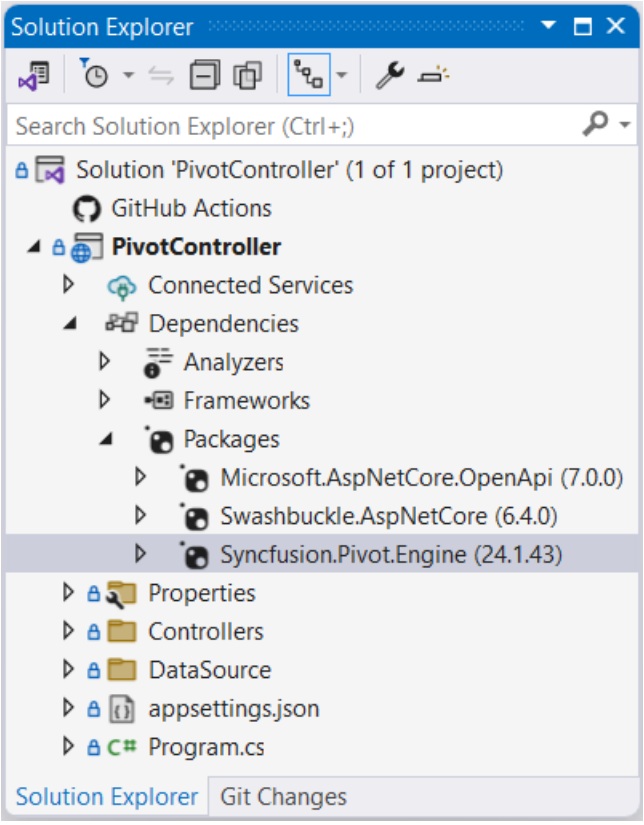
Download and installing Server-side Pivot Engine

1. Download the ASP.NET Core-based stand-alone Pivot Table application from the GitHub repository.

2. The **PivotController** (Server-side) application that is downloaded includes the following files.

- **PivotController.cs** file under **Controllers** folder – This helps to do data communication with Pivot Table.
- **DataSource.cs** file under **DataSource** folder – This file has model classes to define the structure of the data sources.
- The sample data source files **sales.csv** and **sales-analysis.json** under **DataSource** folder.

3. Open the **PivotController** application in Visual Studio where the Syncfusion library Syncfusion.Pivot.Engine will be downloaded automatically from the nuget.org site.



Connecting Pivot Table to Server-side Pivot Engine

1. Run the **PivotController** (Server-side) application which will be hosted in IIS shortly.

2. Then in the Pivot Table sample, set the mode property under dataSourceSettings as **Server** and map the URL of the hosted Server-side application in URL property of dataSourceSettings.

`javascript

<script>

import Vue from "vue";

import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";

Vue.use(PivotViewPlugin);

export default {

data () {

return {

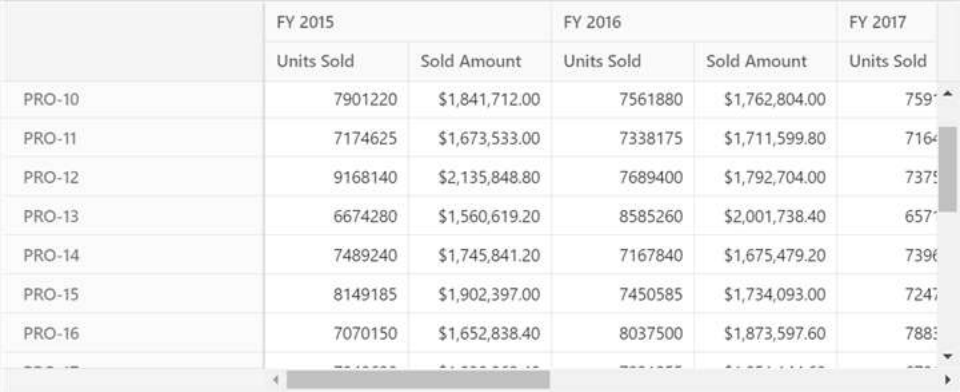
```
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
},
height: 350
}
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
`
```

3. Frame and set the report based on the data source available in the **PivotController** application.

```
`javascript
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data () {
return {
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
rows: [{
name: 'ProductID', caption: 'Product ID'
}],
formatSettings: [{
name: 'Price', format: 'C'
}],
columns: [{
name: 'Year', caption: 'Production Year'
```

```
}],
values: [
{ name: 'Sold', caption: 'Units Sold' },
{ name: 'Price', caption: 'Sold Amount' }
],
},
height: 350
}
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
`
```

4. Run the sample to get the following result.



	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
PRO-10	7901220	$1,841,712.00	7561880	$1,762,804.00	759
PRO-11	7174625	$1,673,533.00	7338175	$1,711,599.80	716
PRO-12	9168140	$2,135,848.80	7689400	$1,792,704.00	737
PRO-13	6674280	$1,560,619.20	8585260	$2,001,738.40	657
PRO-14	7489240	$1,745,841.20	7167840	$1,675,479.20	739
PRO-15	8149185	$1,902,397.00	7450585	$1,734,093.00	724
PRO-16	7070150	$1,652,838.40	8037500	$1,873,597.60	788

Available configurations in Server-side application

Supportive Data Sources

The server-side Pivot Engine supports the following data sources,

- Collection
- JSON
- CSV
- DataTable
- Dynamic

Copyright © 2001 -2024 Syncfusion Inc.

Collection

The collection data sources such as List, IEnumerable, and so on are supported. This can be bound using the **GetData** controller method. Also, in the Pivot Table sample, set the type property under dataSourceSettings to **JSON**, which is also the default enumeration value.

In the server-side application **(PivotController)**, a collection type data source is framed in the **DataSource.cs** file as shown in the following.

```
public class PivotViewData
{
public string ProductID { get; set; }
public string Country { get; set; }
public string Product { get; set; }
public double Sold { get; set; }
public double Price { get; set; }
public string Year { get; set; }
public List<PivotViewData> GetVirtualData()
{
List<PivotViewData> VirtualData = new List<PivotViewData>();
for (int i = 1; i <= 10000; i++)
{
PivotViewData p = new PivotViewData
{
ProductID = "PRO-" + ((100 + i)%20),
Year = (new string[] { "FY 2015", "FY 2016", "FY 2017", "FY 2018", "FY 2019" })[new Random().Next(5)],
Country = (new string[] { "Canada", "France", "Australia", "Germany", "France" })[new Random().Next(5)],
Product = (new string[] { "Car", "Van", "Bike", "Flight", "Bus" })[new Random().Next(5)],
Price = (3.4 * i) + 500,
Sold = (i * 15) + 10
};
VirtualData.Add(p);
}
return VirtualData;
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

`

To bind the data source, set its model type **PivotViewData** to **TValue** of the **PivotEngine** class.

`

```
private PivotEngine<DataSource.PivotViewData> PivotEngine = new
PivotEngine<DataSource.PivotViewData>();
```

`

Then call the data source in **GetData** method of **PivotController.cs** file.

`

```
public async Task<object> GetData(FetchData param)
{
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
async (cacheEntry) =>
{
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here bind the collection type data source.
return new DataSource.PivotViewData().GetVirtualData();
});
}
```

`

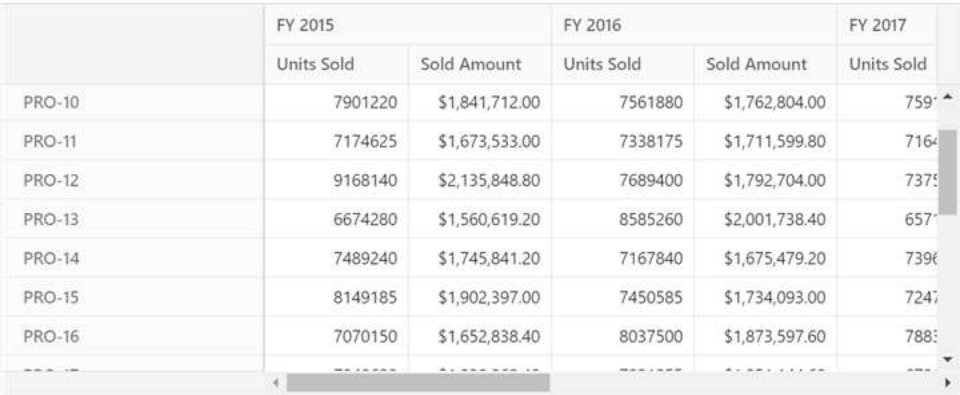
Finally set the appropriate report to the Pivot Table sample based on the above data source.

`javascript

```
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data () {
return {
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
rows: [{
```

Copyright © 2001 -2024 Syncfusion Inc.

```
name: 'ProductID', caption: 'Product ID'
}],
formatSettings: [{
name: 'Price', format: 'C'
}],
columns: [{
name: 'Year', caption: 'Production Year'
}],
values: [
{ name: 'Sold', caption: 'Units Sold' },
{ name: 'Price', caption: 'Sold Amount' }
],
},
height: 350
}
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
`
```



	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
PRO-10	7901220	$1,841,712.00	7561880	$1,762,804.00	759
PRO-11	7174625	$1,673,533.00	7338175	$1,711,599.80	716
PRO-12	9168140	$2,135,848.80	7689400	$1,792,704.00	737
PRO-13	6674280	$1,560,619.20	8585260	$2,001,738.40	657
PRO-14	7489240	$1,745,841.20	7167840	$1,675,479.20	739
PRO-15	8149185	$1,902,397.00	7450585	$1,734,093.00	724
PRO-16	7070150	$1,652,838.40	8037500	$1,873,597.60	788

JSON

The JSON data from a local *.json file type can be connected to the Pivot Table. Here, the file can be read by the **StreamReader** option, which will give the result in the string format. The resultant string needs to be converted to collect data that can be bound to the Server-side pivot engine.

In the Server-side application, **sales-analysis.json** file is available under **DataSource** folder and its model type is defined in **DataSource.cs** file.

```
public class PivotJSONData
{
public string Date { get; set; }
public string Sector { get; set; }
public string EnerType { get; set; }
public string EneSource { get; set; }
public int PowUnits { get; set; }
public int ProCost { get; set; }
public List<PivotJSONData> ReadJSONData(string url)
{
WebClient myWebClient = new WebClient();
Stream myStream = myWebClient.OpenRead(url);
StreamReader stream = new StreamReader(myStream);
string result = stream.ReadToEnd();
stream.Close();
return Newtonsoft.Json.JsonConvert.DeserializeObject<List<PivotJSONData>>(result);
}
}
```

To bind the data source, set its model type **PivotJSONData** to **TValue** of the **PivotEngine** class.

```
private PivotEngine<DataSource.PivotJSONData> PivotEngine = new PivotEngine<DataSource.
PivotJSONData>();
```

Then call the data source in **GetData** method of **PivotController.cs** file.

```
public async Task<object> GetData(FetchData param)
{
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
async (cacheEntry) =>
{
```

Copyright © 2001 -2024 Syncfusion Inc.

```
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here bind JSON type data source from the sales-analysis.json file.
return new DataSource.PivotJSONData().ReadJSONData(_hostingEnvironment.ContentRootPath +
"//DataSource//sales-analysis.json");
});
}
```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```javascript
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data () {
return {
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
type: 'JSON',
rows: [{
name: 'EneSource', caption: 'Energy Source'
}],
formatSettings: [{
name: 'ProCost', format: 'C'
}],
columns: [{
name: 'EnerType', caption: 'Energy Type'
}],
values: [
{ name: 'PowUnits', caption: 'Units Sold' },
{ name: 'ProCost', caption: 'Sold Amount' }
```

```
],
},
height: 350
}
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
`
```

	Biomass		Free Energy		Grand Total
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
Bio-diesel	1042	$1,439.00			1
Ethanol Fuel	595	$1,031.00			
Geo-thermal			1528	$2,115.00	1
Hydro-electric			3378	$3,244.00	3
Solar			7929	$6,210.00	7
Wastage	712	$1,043.00			
Wind			15571	$7,666.00	15

JSON data from any remote server, like a local JSON file, can also be supported. It accepts both directly downloadable files (*.json) and web service URLs. To bind this, the URL of the* .json file of a remote server has to be mapped under the **GetData** method. The rest of the configurations are the same as described above.

In the server-side application, the CDN link is used to connect the same **sales-analysis.json** file which is already hosted in the Syncfusion server.

```
`
public async Task<object> GetData(FetchData param)
{
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
async (cacheEntry) =>
{
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here bind JSON type data source from remote server.
```

```
return new DataSource.PivotJSONData().ReadJSONData("http://cdn.syncfusion.com/data/sales-analysis.json");
});
}
`
```

CSV

The CSV data from a local *.csv file type can be connected to the Pivot Table. Here, the file can be read by the **StreamReader** option, which will give the result in the string format. The resultant string needs to be converted to collect data that can be bound to the server-side pivot engine. Also, in the Pivot Table sample, set the type property under dataSourceSettings as **CSV**.

In the server application, the **sales.csv** file is available under the **DataSource** folder, and its model type is defined in the **DataSource.cs** file.

```
`
public class PivotCSVData
{
public string Region { get; set; }
public string Country { get; set; }
public string ItemType { get; set; }
public string SalesChannel { get; set; }
public string OrderPriority { get; set; }
public string OrderDate { get; set; }
public int OrderID { get; set; }
public string ShipDate { get; set; }
public int UnitsSold { get; set; }
public double UnitPrice { get; set; }
public double UnitCost { get; set; }
public double TotalRevenue { get; set; }
public double TotalCost { get; set; }
public double TotalProfit { get; set; }
public List<string[]> ReadCSVData(string url)
{
List<string[]> data = new List<string[]>();
using (StreamReader reader = new StreamReader(new WebClient().OpenRead(url)))
{
string line;
```

```
while ((line = reader.ReadLine()) != null)
{
line = line.Trim();
if (!string.IsNullOrWhiteSpace(line))
{
data.Add(line.Split(','));
}
}
return data;
}
}
}
```

To bind the data source, set its model type **PivotCSVData** to **TValue** of the **PivotEngine** class.

```
private PivotEngine<DataSource.PivotCSVData> PivotEngine = new PivotEngine<DataSource.
PivotCSVData>();
```

Then call the data source in **GetData** method of **PivotController.cs** file.

```
public async Task<object> GetData(FetchData param)
{
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
async (cacheEntry) =>
{
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here bind CSV type data source from sales.csv file.
return new DataSource.PivotCSVData().ReadCSVData(_hostingEnvironment.ContentRootPath +
"//DataSource//sales.csv");
});
}
```

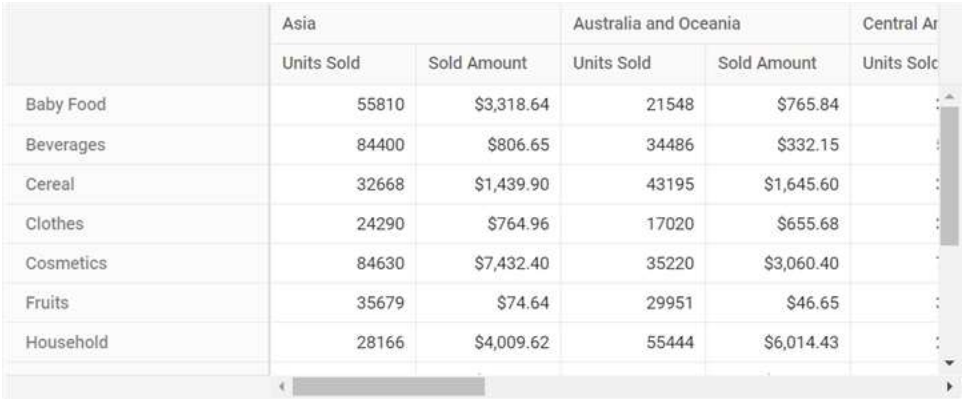
Finally set the appropriate report to the Pivot Table sample based on the above data source.

Copyright © 2001 -2024 Syncfusion Inc.

```
`javascript
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data () {
return {
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
type: 'CSV',
rows: [{
name: 'ItemType', caption: 'Item Type'
}],
formatSettings: [{
name: 'UnitPrice', format: 'C'
}],
columns: [{
name: 'Region'
}],
values: [
{ name: 'UnitsSold', caption: 'Units Sold' },
{ name: 'UnitPrice', caption: 'Sold Amount' }
],
},
height: 350
}
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
```

</style>

`



	Asia		Australia and Oceania		Central Ar
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
Baby Food	55810	$3,318.64	21548	$765.84	
Beverages	84400	$806.65	34486	$332.15	
Cereal	32668	$1,439.90	43195	$1,645.60	
Clothes	24290	$764.96	17020	$655.68	
Cosmetics	84630	$7,432.40	35220	$3,060.40	
Fruits	35679	$74.64	29951	$46.65	
Household	28166	$4,009.62	55444	$6,014.43	

CSV data from any remote server, like a local CSV file, can also be supported. It accepts both directly downloadable files (*.csv) and web service URLs. To bind this, the URL of the* .csv file of a remote server has to be mapped under **GetData** method. The rest of the configurations are the same as described above.

In the server application, the CDN link is used to connect the same **sales.csv** file which is already hosted in the Syncfusion server.

`

public async Task<object> GetData(FetchData param)

{

return await _cache.GetOrCreateAsync("dataSource" + param.Hash,

async (cacheEntry) =>

{

cacheEntry.SetSize(1);

cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);

// Here bind CSV type data source from remote server.

return new DataSource.PivotCSVData().ReadCSVData("http://cdn.syncfusion.com/data/sales-analysis.csv");

});

}

`

DataTable

In the server-side application, there is a manually created DataTable **BusinessObjectsDataView** by mapping the model type **PivotViewData** in **DataSource.cs** file.

`

public class BusinessObjectsDataView

```
{
public DataTable GetDataTable()
{
DataTable dt = new DataTable("BusinessObjectsDataTable");
PropertyDescriptorCollection pdc = TypeDescriptor.GetProperties(typeof(PivotViewData));
foreach (PropertyDescriptor pd in pdc)
{
dt.Columns.Add(new DataColumn(pd.Name, pd.PropertyType));
}
List<PivotViewData> list = new PivotViewData().GetVirtualData();
foreach (PivotViewData bo in list)
{
DataRow dr = dt.NewRow();
foreach (PropertyDescriptor pd in pdc)
{
dr[pd.Name] = pd.GetValue(bo);
}
dt.Rows.Add(dr);
}
return dt;
}
}
```

To bind the data source, set its model type **PivotViewData** to **TValue** of the **PivotEngine** class.

```
private PivotEngine<DataSource.PivotViewData> PivotEngine = new
PivotEngine<DataSource.PivotViewData>();
```

Then call the data source in **GetData** method of **PivotController.cs** file.

```
public async Task<object> GetData(FetchData param)
{
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
```

Copyright © 2001 -2024 Syncfusion Inc.

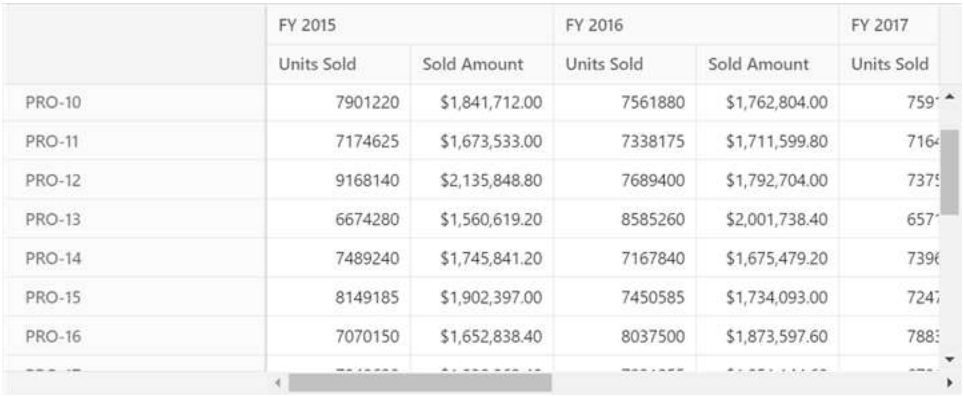
```
async (cacheEntry) =>
{
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here bind the DataTable.
return new DataSource.BusinessObjectsDataView().GetDataTable();
});
}
`
```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```javascript
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data () {
return {
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
rows: [{
name: 'ProductID', caption: 'Product ID'
}],
formatSettings: [{
name: 'Price', format: 'C'
}],
columns: [{
name: 'Year', caption: 'Production Year'
}],
values: [
{ name: 'Sold', caption: 'Units Sold' },
{ name: 'Price', caption: 'Sold Amount' }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
],
},
height: 350
}
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
`
```



	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
PRO-10	7901220	$1,841,712.00	7561880	$1,762,804.00	759
PRO-11	7174625	$1,673,533.00	7338175	$1,711,599.80	716
PRO-12	9168140	$2,135,848.80	7689400	$1,792,704.00	737
PRO-13	6674280	$1,560,619.20	8585260	$2,001,738.40	657
PRO-14	7489240	$1,745,841.20	7167840	$1,675,479.20	739
PRO-15	8149185	$1,902,397.00	7450585	$1,734,093.00	724
PRO-16	7070150	$1,652,838.40	8037500	$1,873,597.60	788

Dynamic

The model type has to be defined in the aforementioned data sources. However, there is no need to define a model type for the following data sources, which are also supported by the server-side pivot engine.

ExpandoObject

In the server-side application, an **ExpandoObject** type data source is available under the class **PivotExpandoData** in **DataSource.cs** file.

```
`
public class PivotExpandoData
{
public List<ExpandoObject> Orders { get; set; } = new List<ExpandoObject>();
public List<ExpandoObject> GetExpandoData()
{
Orders = Enumerable.Range(1, 75).Select((x) =>
{
dynamic d = new ExpandoObject();
```

Copyright © 2001 -2024 Syncfusion Inc.

```
d.OrderID = 1000 + (x % 100);
d.CustomerID = (new string[] { "ALFKI", "ANANTR", "ANTON", "BLONP", "BOLID" })[new Random().Next(5)];
d.Freight = (new double[] { 2, 1, 4, 5, 3 })[new Random().Next(5)] * x;
d.OrderDate = (new DateTime[] { new DateTime(2010, 11, 5), new DateTime(2018, 10, 3), new DateTime(1995, 9, 9), new DateTime(2012, 8, 2), new DateTime(2015, 4, 11) })[new Random().Next(5)];
d.ShipCountry = (new string[] { "USA", "UK" })[new Random().Next(2)];
d.Verified = (new bool[] { true, false })[new Random().Next(2)];
return d;
}).Cast<ExpandoObject>().ToList<ExpandoObject>();
return Orders;
}
}
```

To bind the data source, set its model type as **ExpandoObject** to **TValue** of the **PivotEngine** class.

```
private PivotEngine<ExpandoObject> PivotEngine = new PivotEngine<ExpandoObject>();
```

Then call the data source in **GetData** method of **PivotController.cs** file.

```
public async Task<object> GetData(FetchData param)
{
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
async (cacheEntry) =>
{
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here returns ExpandoObject type data source.
return new DataSource.PivotExpandoData().GetExpandoData();
});
}
```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```javascript
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data () {
return {
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
rows: [{
name: 'CustomerID', caption: 'Customer ID'
}],
columns: [{
name: 'ShipCountry', caption: 'Ship Country'
}],
values: [
{ name: 'Freight', caption: 'Units Sold' }
],
},
height: 350
}
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
`
```

Copyright © 2001 -2024 Syncfusion Inc.

	UK	USA	Grand Total
	Units Sold	Units Sold	Units Sold
ALFKI	1187	738	1925
ANANTR	1011	814	1825
ANTON	1046	632	1678
BLONP	1080	749	1829
BOLID	492	929	1421
Grand Total	4816	3862	8678

Dynamic Objects

In the server-side application, a data source is framed by dynamic objects which is available under the class **PivotDynamicData** in the **DataSource.cs** file.

```
public class PivotDynamicData
{
public List<DynamicDictionary> Orders = new List<DynamicDictionary>() { };
public List<DynamicDictionary> GetDynamicData()
{
Orders = Enumerable.Range(1, 100).Select((x) =>
{
dynamic d = new DynamicDictionary();
d.OrderID = 100 + x;
d.CustomerID = (new string[] { "ALFKI", "ANANTR", "ANTON", "BLONP", "BOLID" })[new Random().Next(5)];
d.Freight = (new double[] { 2, 1, 4, 5, 3 })[new Random().Next(5)] * x;
d.OrderDate = (new DateTime[] { new DateTime(2010, 11, 5), new DateTime(2018, 10, 3), new DateTime(1995, 9, 9), new DateTime(2012, 8, 2), new DateTime(2015, 4, 11) })[new Random().Next(5)];
d.ShipCountry = (new string[] { "USA", "UK" })[new Random().Next(2)];
d.Verified = (new bool[] { true, false })[new Random().Next(2)];
return d;
}).Cast<DynamicDictionary>().ToList<DynamicDictionary>();
return Orders;
}
public class DynamicDictionary : System.Dynamic.DynamicObject
{
Dictionary<string, object> dictionary = new Dictionary<string, object>();
public override bool TryGetMember(GetMemberBinder binder, out object result)
```

```
{
string name = binder.Name;
return dictionary.TryGetValue(name, out result);
}
public override bool TrySetMember(SetMemberBinder binder, object value)
{
dictionary[binder.Name] = value;
return true;
}
//The "GetDynamicMemberNames" method of the "DynamicDictionary" class must be overridden and return the property names to perform data operation and editing while using dynamic objects.
public override System.Collections.Generic.IEnumerable<string> GetDynamicMemberNames()
{
return this.dictionary?.Keys;
}
}
}
```

To bind the data source, set its class **PivotDynamicData** to **TValue** of the **PivotEngine** class.

```
private PivotEngine<DataSource.PivotDynamicData> PivotEngine = new PivotEngine<DataSource.PivotDynamicData>();
```

Then call the data source in **GetData** method of **PivotController.cs** file.

```
public async Task<object> GetData(FetchData param)
{
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
async (cacheEntry) =>
{
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here bind data source with dynamic objects.
return new DataSource.PivotDynamicData().GetDynamicData();
```

Copyright © 2001 -2024 Syncfusion Inc.

```
});
}
`
```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```
`javascript
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
export default {
data () {
return {
dataSourceSettings: {
url: 'https://localhost:44350/api/pivot/post',
mode: 'Server',
rows: [{
name: 'CustomerID', caption: 'Customer ID'
}],
columns: [{
name: 'ShipCountry', caption: 'Ship Country'
}],
values: [
{ name: 'Freight', caption: 'Units Sold' }
],
},
height: 350
}
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

`

	UK	USA	Grand Total
	Units Sold	Units Sold	Units Sold
ALFKI	1030	742	1772
ANANTR	1352	1018	2370
ANTON	2525	1782	4307
BLONP	1786	1002	2788
BOLID	1409	2012	3421
Grand Total	8102	6556	14658

Controller Configuration

Memory Cache

In the server-side application, the Memory Cache option is used to store the data source and engine properties in RAM, which will be used for UI operations. To improve performance, this limits the execution of all initial rendering code to regenerate the aggregated values during each UI operation. The codes below show how we use the memory cache option in the **GetEngine** method to store engine properties.

`

```
public async Task<EngineProperties> GetEngine(FetchData param)
{
isRendered = false;
// Engine properties are stored in memory cache with GUID "parem.Hash".
return await _cache.GetOrCreateAsync("engine" + param.Hash,
async (cacheEntry) =>
{
isRendered = true;
cacheEntry.SetSize(1);
// Memory cache expiration time can be set here.
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
PivotEngine.Data = await GetData(param);
return await PivotEngine.GetEngine(param);
});
}
```

`

The engine properties are stored in RAM as a cache with a unique ID (GUID) that is transferred from the client-side source code. The GUID is generated at random and will be changed if the page containing the Pivot Table is refreshed or opened in a new tab/window. As a result, each GUID's memory cache contains unique information, and the component operates independently.

Copyright © 2001 -2024 Syncfusion Inc.

Meanwhile, the memory cache is set to expire after 60 minutes from RAM to free its memory. If the component is still running, the data will be generated and stored for another 60 minutes.

Methods and its needs

- **Post:** Allows to get the information from the client-side source and calls appropriate controller methods.
- **GetEngine:** Allows to store the engine properties in RAM as a cache which fires on initial rendering or when the memory cache is expired.
- **GetData:** Allows to store data source in RAM as a cache which fires on initial rendering or when the memory cache is expired.
- **GetMembers:** Allows to get the members of a field. This fires when the member editor is opened to do a filtering operation.
- **GetRawData:** Allows to get raw data of an aggregated value cell. This fires when the drill-through or editing dialog is opened.
- **GetPivotValues:** Allows to update the stored engine properties in-memory cache and returns the aggregated values to browser to render the Pivot Table. Here, the return value can be modified. The Pivot Table will be rendered browser-based on this.

Pivot chart in Vue Pivotview component

In pivot table component, pivot chart would act as an additional visualization component with its basic and important characteristic like drill down and drill up, 15+ chart types, series customization, axis customization, legend customization, export, print and tooltip. Its main purpose is to show the pivot data in graphical format.

If user prefers, the pivot chart component can also be displayed individually with pivot values and can change the report dynamically with the help of field list and grouping bar. Using the displayOption property in pivot table, user can set the visibility of grid and chart in pivot table component. It holds below properties,

- view: Specifies the pivot table component to display grid alone or chart alone or both.
- primary: Specifies the pivot table to display either grid or chart as primary component during initial loading. It is applicable only when setting the property view to **Both**.

You should inject the PivotChart module to make the its features available in the pivot table.

The below sample displays the pivot chart component based on the pivot report bound on it.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 240,
      displayOption: { view: 'Chart' },
      chartSettings: { chartSeries: { type: 'Column' } }
    }
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs177" %}

Data binding

End user can bind both local and remote data binding options available in the component to feed the data. The dataSource property can be assigned either with an instance of DataManager or JavaScript object array collection.

For more information refer here.

Chart Types

Supports 21 different types of charts as follows,

- Line
- Column
- Area
- Bar
- StepArea
- StackingLine
- StackingColumn
- StackingArea
- StackingBar
- StepLine
- Pareto
- Bubble
- Scatter

Copyright © 2001 -2024 Syncfusion Inc.

- Spline
- SplineArea
- StackingLine100
- StackingColumn100
- StackingBar100
- StackingArea100
- Polar
- Radar

Line is the default pivot chart type. User can change the pivot chart type by using the property type in chartSeries.

In the below code sample, the pivot chart type is set as **Bar**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: { chartSeries: { type: 'Bar' } }
    }
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs178" %}

Accumulation Charts

Supports 4 different types of accumulation charts as follows,

- Pie
- Doughnut
- Funnel
- Pyramid

As like other chart types it can be changed using the property type in chartSeries.

In the below code sample, the **Pie** chart is rendered, and the other accumulation charts can be switched using the drop-down list.

APP.VUE

```
<template>
    <div class="control-section" id="pivot-grid-section">
        <div id="dropdown-control" style="margin-bottom:5px;">
            <table style="width: 350px;margin-left: 50;">
                <tbody>
                    <tr style="height: 50px">
                        <td>
                            <div>
                                <b>Accumulation Chart:</b>
                            </div>
                        </td>
                        <td>
                            <div>
                                <ejs-dropdownlist id="dropdownlist"
:change="ddlOnChange" :dataSource="chartTypes"
                                    index="0"></ejs-dropdownlist>
                            </div>
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
        <div class="content-wrapper" id="app" style="height: 250px;">
            <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
                :chartSettings="chartSettings"
:displayOption="displayOption"></ejs-pivotview>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { DropDownListPlugin, ChangeEventArgs } from "@syncfusion/ej2-vue-
dropdowns";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
```

```
Vue.use(DropDownListPlugin)
export default {
    data() {
        return {
            dataSourceSettings: {
                dataSource: pivotData,
                expandAll: false,
                columns: [{ name: 'Year', caption: 'Production Year' }, {
name: 'Quarter' }],
                values: [{ name: 'Sold', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
                rows: [{ name: 'Country' }, { name: 'Products' }],
                formatSettings: [{ name: 'Amount', format: 'C0' }],
                filters: []
            },
            height: 350,
            displayOption: { view: 'Chart' },
            chartSettings: { chartSeries: { type: 'Pie' } },
            chartTypes: ["Pie", "Doughnut", "Pyramid", "Funnel"]
        }
    },
    methods: {
        ddlOnChange: function (args) {
            let pivotObj = (this.$refs.pivotview).ej2Instances;
            pivotObj.chartSettings.chartSeries.type = args.value.toString();
        }
    },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs179" %}

Drill Down/Up

In the accumulation charts, drill down and drill up operations can be performed using the built-in context menu option. It will be shown while clicking on the chart series. The context menu has the following options:

Expand - It is to drill down the corresponding series until the last level.

Collapse - It is to drill up the corresponding series until the first level.

Exit - It is to close the context menu.

The drill operation in accumulation charts can be performed only for row headers.

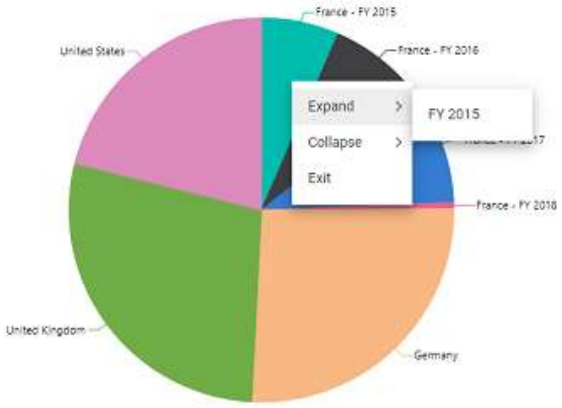
APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption"
            :chartSettings="chartSettings" :height="height"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
    data() {
        return {
            dataSourceSettings: {
                dataSource: pivotData,
                expandAll: false,
                columns: [{ name: 'Products' }],
                values: [{ name: 'Sold', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
                rows: [{ name: 'Country' }, { name: 'Year', caption:
'Production Year' }, { name: 'Quarter' }],
                formatSettings: [{ name: 'Amount', format: 'C0' }],
                filters: []
            },
            height: 350,
            displayOption: { view: 'Chart' },
            chartSettings: { chartSeries: { type: 'Pie' } }
        }
    },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs180" %}

Copyright © 2001 -2024 Syncfusion Inc.



Column Headers and Delimiters

Unlike other chart types, the accumulation charts consider the values of a single column from the pivot table to be drawn. Preferably the first column of the pivot table is considered by default. But it can be changed by defining the column headers using the columnHeader property in chartSettings.

If the column has more than one header, then need to mention all the headers separated by the delimiter **-**, for example,**Germany-Road Bikes**. Using the property columnDelimiter in chartSettings, one can set the desired delimiter to separate the column headers.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption"
            :chartSettings="chartSettings" :height="height"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
    data() {
        return {
            dataSourceSettings: {
                dataSource: pivotData,
                expandAll: false,
                drilledMembers: [{ name: 'Country', items: ['Germany'] }],
                columns: [{ name: 'Country' }, { name: 'Products' }],
                values: [{ name: 'Sold', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
                rows: [{ name: 'Year' }, { name: 'Quarter' }],
                formatSettings: [{ name: 'Amount', format: 'C0' }],
                filters: []
            },
            height: 350,
```

```
            displayOption: { view: 'Chart' },
            chartSettings: { columnHeader: "Germany-Road Bikes",
columnDelimiter: '-', chartSeries: { type: 'Doughnut' } }
        }
    },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs181" %}

Label Customization

The data labels are visible by default showing header name. Its visibility can be modified using the visible boolean property in dataLabel. With regard to the label arrangement, the **Smart Labels** options help to arrange labels efficiently without overlapping. It can be disabled by setting the enableSmartLabels property in chartSettings as "false".

The position property in dataLabel allows to specify the position of the data label. The available options are,

- Outside: Positions the label outside the point. It is the default option.
- Inside: Positions the label inside the point.

In the following code sample, the data labels are placed inside.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption"
            :chartSettings="chartSettings" :height="height"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
    data() {
        return {
            dataSourceSettings: {
                dataSource: pivotData,
                expandAll: false,
```

```
                columns: [{ name: 'Year', caption: 'Production Year' }, {
name: 'Quarter' }],
                values: [{ name: 'Sold', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
                rows: [{ name: 'Country' }, { name: 'Products' }],
                formatSettings: [{ name: 'Amount', format: 'C0' }],
                filters: []
            },
            height: 350,
            displayOption: { view: 'Chart' },
            chartSettings: {
                enableSmartLabels: false,
                chartSeries: { dataLabel: {visible:true, position: 'Inside'
}, type: 'Pyramid' }
            }
        }
    },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs182" %}

The **Connector Line** will be visible when the data label is placed outside the chart. It can be customized using the connectorStyle property in dataLabel for its color, length, width etc. In the following code sample, the connector line is customized.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption"
            :chartSettings="chartSettings" :height="height"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
    data() {
        return {
            dataSourceSettings: {
                dataSource: pivotData,
                expandAll: false,
```

```
                columns: [{ name: 'Year', caption: 'Production Year' }, {
name: 'Quarter' }],
                values: [{ name: 'Sold', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
                rows: [{ name: 'Country' }, { name: 'Products' }],
                formatSettings: [{ name: 'Amount', format: 'C0' }],
                filters: []
            },
            height: 350,
            displayOption: { view: 'Chart' },
            chartSettings: {
                chartSeries: {
                    dataLabel: {visible:true, position: 'Outside',
connectorStyle: { length: '50px', width: 2, dashArray: '5,3', color:
'#f4429e' } },
                    type: 'Funnel'
                }
            }
        }
    },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs183" %}

Pie and Doughnut Customization

User can draw pie and doughnut charts within the specified range using the startAngle and endAngle properties in chartSeries. The default value of the startAngle property is **0**, and the endAngle property is **360**. By customizing these properties, user can draw semi pie and semi doughnut charts.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption"
            :chartSettings="chartSettings" :height="height"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
    data() {
```

```
        return {
            dataSourceSettings: {
                dataSource: pivotData,
                expandAll: false,
                columns: [{ name: 'Year', caption: 'Production Year' }, {
name: 'Quarter' }],
                values: [{ name: 'Sold', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
                rows: [{ name: 'Country' }, { name: 'Products' }],
                formatSettings: [{ name: 'Amount', format: 'C0' }],
                filters: []
            },
            height: 350,
            displayOption: { view: 'Chart' },
            chartSettings: { chartSeries: { startAngle: 270, endAngle: 90,
type: 'Doughnut' } }
        }
    },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs184" %}

Users can get doughnut chart from pie chart and vice-versa using the innerRadius property in chartSeries. If the property is greater than **0** percent, the doughnut chart will appear from the pie chart.

It takes the value only in percentage.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption"
            :chartSettings="chartSettings" :height="height"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
    data() {
        return {
            dataSourceSettings: {
                dataSource: pivotData,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                expandAll: false,
                columns: [{ name: 'Year', caption: 'Production Year' }, {
name: 'Quarter' }],
                values: [{ name: 'Sold', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
                rows: [{ name: 'Country' }, { name: 'Products' }],
                formatSettings: [{ name: 'Amount', format: 'C0' }],
                filters: []
            },
            height: 350,
            displayOption: { view: 'Chart' },
            chartSettings: { chartSeries: { innerRadius: '140', type: 'Pie'
} }
        }
    },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs185" %}

Exploding Series Points

Exploding can be enabled by setting the explode property in chartSeries to **true**. The series points will be exploded either on mouse click or touch.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption"
            :chartSettings="chartSettings" :height="height"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
    data() {
        return {
            dataSourceSettings: {
                dataSource: pivotData,
                expandAll: false,
                columns: [{ name: 'Year', caption: 'Production Year' }, {
name: 'Quarter' }],
```

```
                values: [{ name: 'Sold', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
                rows: [{ name: 'Country' }, { name: 'Products' }],
                formatSettings: [{ name: 'Amount', format: 'C0' }],
                filters: []
            },
            height: 350,
            displayOption: { view: 'Chart' },
            chartSettings: { chartSeries: { explode: true, type: 'Pie' } }
        }
    },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs186" %}

Field List

User can enable the field list by setting the property showFieldList in pivot table as **true**.

By using this, user can customize the report dynamically and view the result in pivot chart. For more information regarding the field list, refer the field list topic.

In the following sample, the Popup mode of field list is enabled in the pivot chart.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:showFieldList="showFieldList" :displayOption="displayOption"
:chartSettings="chartSettings" :height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart, FieldList } from "@syncfusion/ej2-vue-
pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
```

```
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: { chartSeries: { type: 'Column' }},
      showFieldList: true
    }
  },
    provide: {
        pivotview: [PivotChart, FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs187" %}

Grouping Bar

User can enable the grouping bar by setting the property showGroupingBar in pivot table as **true**. The grouping bar in pivot chart shows a dropdown list in value axis instead of buttons. The dropdown list holds list of value fields bounded in the dataSourceSettings and it can be switched to draw the pivot chart with the selected value field. This has been defined as the default behavior in the pivot chart component. For more information regarding the grouping bar, refer the grouping bar topic.

For multiple axis support, buttons will be placed in value axis instead of dropdown list.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:showGroupingBar="showGroupingBar" :displayOption="displayOption"
:chartSettings="chartSettings" :height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart, GroupingBar } from "@syncfusion/ej2-
vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
```

```
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 240,
      displayOption: { view: 'Chart' },
      chartSettings: { chartSeries: { type: 'Column' }},
      showGroupingBar: true
    }
  },
    provide: {
        pivotview: [PivotChart, GroupingBar]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs188" %}

For accumulation charts alone, a drop-down list will be placed in the column axis instead of the buttons. The drop-down list shows the column headers available in the pivot table. Users can dynamically switch column headers with the help of the drop-down list, and the accumulation chart will be updated accordingly.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:showGroupingBar="showGroupingBar"
            :displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart, GroupingBar } from "@syncfusion/ej2-
vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
    data() {
        return {
            dataSourceSettings: {
                dataSource: pivotData,
                expandAll: false,
                columns: [{ name: 'Year', caption: 'Production Year' }, {
name: 'Quarter' }],
                values: [{ name: 'Sold', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
                rows: [{ name: 'Country' }, { name: 'Products' }],
                formatSettings: [{ name: 'Amount', format: 'C0' }],
```

```
                filters: []
            },
            height: 240,
            displayOption: { view: 'Chart' },
            chartSettings: { chartSeries: { type: 'Pie' } },
            showGroupingBar: true
        }
    },
    provide: {
        pivotview: [PivotChart, GroupingBar]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs189" %}

Single Axis

By default, the pivot chart will be drawn with the value field (measure) which is set first in the report under value axis. But, user can change to specific value field using the property value in chartSettings.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
```

```
      chartSettings: { value: 'Amount', chartSeries: { type: 'Column' } }
    }
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs190" %}

Multiple Axis

User can draw the pivot chart with multiple value fields by setting the property enableMultipleAxis to **true**. In the following sample, the pivot chart will be drawn with both the “Sold” and “Amount” value fields available in the data source.

The multiple axis support is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: { enableMultipleAxis: true, chartSeries: { type:
'Column' } }
    }
  },
```

```
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs191" %}

If the user binds more value fields, the result will be multiple pivot charts, and each chart will shrink within the parent container height. To avoid this, set the enableScrollOnMultiAxis property in chartSettings to **true**. By doing so, each pivot chart will only shrink to a minimal "160px" - "180px" height showing a vertical scrollbar for a clear view.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' },{name: 'Products', type:'Count'}],
        rows: [{ name: 'Country' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: { enableScrollOnMultiAxis:true,
      enableMultipleAxis: true, chartSeries: { type: 'Column' } }
    }
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
```

```
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs192" %}

Meanwhile, there is another way to display multiple values in a chart. In this approach, the series drawn from multiple values are grouped and displayed in a single chart. And, based on the values, multiple Y axis scales will be framed with different ranges. This can be achieved by setting the properties enableMultipleAxis as **true** and multipleAxisMode as Single in chartSettings.

In the following code sample, the pivot chart can be seen as a single chart with multiple value fields such as **Sold** and **Amount** that are drawn as multiple Y axis.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: { enableMultipleAxis: true, multipleAxisMode :
'Single', chartSeries: { type: 'Column' } }
    }
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs193" %}

Additionally, to display chart series for multiple values within a single y-axis, set the properties enableMultipleAxis to **true** and the multipleAxisMode to **Combined**, in the chartSettings.

The y-axis range values will be formatted using the first value field on the value axis. For example, if the first value field is in currency format and the remaining value fields are in different number formats or no format, the y-axis range values will be displayed in the currency format of the first value field.

The pivot chart in the following code sample can be seen as a single chart with multiple value fields such as **Sold** and **Amount** drawn as a single y-axis.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: { enableMultipleAxis: true, multipleAxisMode:
'Combined', chartSeries: { type: 'Column' } }
    }
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs194" %}

Copyright © 2001 -2024 Syncfusion Inc.

Show point color based on members

When multiple axes are enabled, you can display the same color for each member in the column axis by setting the showPointColorByMembers property to **true** in the chartSettings. As a result, the end user can easily identify each member across different measures in the entire chart.

Furthermore, end user can see or hide specific members across different measures in the entire chart with a single click on the legend item.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: {  title: 'Sales Analysis', value: 'Amount',
chartSeries: { type: 'Column' },
        enableMultipleAxis: true, showPointColorByMembers: true,
multipleAxisMode: 'Stacked',
        primaryYAxis: {border: {width: '0'}} }
    }
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs195" %}

Copyright © 2001 -2024 Syncfusion Inc.

Series customization

User can customize series of the pivot chart using chartSeries in chartSettings. The changes handled in the property will be reflected commonly in all chart series.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: { chartSeries: { type: 'Column', enableTooltip: false,
border: { color: '#000', width: 2 } } }
    }
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs196" %}

User can also customize the pivot chart series individually using the chartSeriesCreated event, which occurs after the pivot chart series has been created. You can customize each series individually by iterating them.

In the following sample, the even series are hidden in the pivot chart.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSeriesCreated="chartSeriesCreated"
:chartSettings="chartSettings" :height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart, ChartSeriesCreatedEventArgs } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: { chartSeries: { type: 'Column' } }
    }
  },
  methods: {
    chartSeriesCreated: function(args: ChartSeriesCreatedEventArgs) {
        for (let pos:number = 0; pos < args.series.length; pos++) {
            if (pos % 2 == 0) {
                args.series[pos].visible = false;
            }
        }
    },
  }
  provide: {
    pivotview: [PivotChart]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs197" %}

Axis Customization

User can customize axis of the pivot chart using primaryXAxis and primaryYAxis properties in chartSettings.

In the following sample, title of y-axis and x-axis are customized.

Axis customization is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: {
        chartSeries: { type: 'Column' },
        primaryXAxis: { title: 'X axis title'},
        primaryYAxis: { title: 'Y axis title' }
      }
    }
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs198" %}

Legend customization

User can customize legend using legendSettings in chartSettings. By default, legend will be visible and it can be hidden by setting the property visible in legendSettings as **false**.

The pivot chart support different types of legend shapes as follows,

- Circle
- Rectangle
- VerticalLine
- Pentagon
- InvertedTriangle
- SeriesType
- Triangle
- Diamond
- Cross
- HorizontalLine

Here **SeriesType** would act as the default shape and it can changed using the property LegendShape in chartSeries.

Also user can set the position of the legend in pivot chart using the property position in legendSettings. The available options to set the legend position are as follows,

- Auto: Places the legend based on area type. This is the default.
- Top: Displays the legend at the top of the pivot chart.
- Left: Displays the legend at the left of the pivot chart.
- Bottom: Displays the legend at the bottom of the pivot chart.
- Right: Displays the legend at the right of the pivot chart.
- Custom: Displays the legend based on the given x and y values.

By default, the legend is not visible for the accumulation chart types like pie, doughnut, pyramid, and funnel.

In the following sample, the legend shape and its position can be customized.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
```

```
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: {
        legendSettings: { position: 'Right' },
        chartSeries: { type: 'Column', legendShape: 'Pentagon' }
      }
    }
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs199" %}

User interaction

Marker and crossHair

User can enable and customize the marker and crosshair using marker and crosshair properties in chartSettings respectively.

Also user can enable and customize the crosshair tooltip for axes using crosshairTooltip.

Marker and crosshair is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

In the following sample, the marker and crosshair can be enabled and customized.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: {
        crosshair: { enable: true },
        chartSeries: {
         type: 'Line',
         marker: { fill: '#EEE', height: 10, width: 10, shape: 'Pentagon',
visible: true }
        },
        primaryXAxis: { crosshairTooltip: { enable: true, fill: '#ff0000' }
},
        primaryYAxis: { crosshairTooltip: { enable: true, fill: '#0000FF' }
}
      }
    }
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs200" %}

Zooming and panning

User can customize zooming and panning option using the property zoomSettings in chartSettings.

The pivot chart support four types of zooming which can be set as follows,

- enablePinchZooming
- enableSelectionZooming
- enableDeferredZooming
- enableMouseWheelZooming

and three modes of zooming direction that specifies whether to zoom vertically or horizontally or in both ways which are,

- x: Pivot chart can be zoomed horizontally.
- y: Pivot chart can be zoomed vertically.
- x,y: Pivot chart can be zoomed both vertically and horizontally.

Copyright © 2001 -2024 Syncfusion Inc.

This can be set using the property mode in zoomSettings. By default, if the pivot chart is zoomed, a toolbar would display with the options - Zoom, ZoomIn, ZoomOut, Pan, Reset. User can also customize its option using the property toolbarItems in zoomSettings.

Zooming and panning is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

In the following sample, all the four types of zooming are enabled with toolbar options.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: {
        chartSeries: {
            type: 'Column'
        },
        zoomSettings: {
            enableDeferredZooming: true,
            enableMouseWheelZooming: true,
            enablePinchZooming: true,
            enableSelectionZooming: true
        }
      }
    }
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
```

```
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs201" %}

Tooltip

By default, tooltip for the pivot chart is enabled. User can customize it by using the property tooltip in chartSettings.

The tooltip can be disabled by setting the property enable in tooltip as **false**.

In the following sample, the default appearance of tooltip is modified.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: {
        chartSeries: {
         type: 'Column'
        },
        tooltip: {
            enableMarker: true,
            textStyle: { color: '#000' },
            fill: '#FFF',
            opacity: 1,
            border: { color: '#000' }
        }
      }
    }
```

```
  },
    provide: {
        pivotview: [PivotChart]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs202" %}

Export

The pivot chart can be exported using the chartExport method which holds parameters like export type, file name, PDF orientation, width, and height in the same order. The mandatory parameters for this method are export type and file name whereas other parameters are optional.

The following are the four export types:

- PNG
- JPEG
- SVG
- PDF

In the following sample, exporting can be done using an external button named as "Export".

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="chartexport" :isPrimary="isPrimary" v-
on:click.native="exportClick">Export</ejs-button>
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :displayOption="displayOption"
:chartSettings="chartSettings" :height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
```

```
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: {
        chartSeries: {
         type: 'Column'
        }
      },
      isPrimary: true
    }
  },
  methods: {
    exportClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.chartExport('PNG', 'result');
    }
  }
  provide: {
        pivotview: [PivotChart]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs203" %}

Print

The rendered pivot chart can be printed directly from the browser by calling printChart method.

In the following sample, printing can be done using an external button named as "Print".

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="chartprint" :isPrimary="isPrimary" v-
on:click.native="printClick">Print</ejs-button>
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :displayOption="displayOption"
:chartSettings="chartSettings" :height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotChart } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
```

```
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view: 'Chart' },
      chartSettings: {
        chartSeries: {
         type: 'Column'
        }
      },
      isPrimary: true
    }
  },
  methods: {
    printClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.printChart();
    }
  }
  provide: {
        pivotview: [PivotChart]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

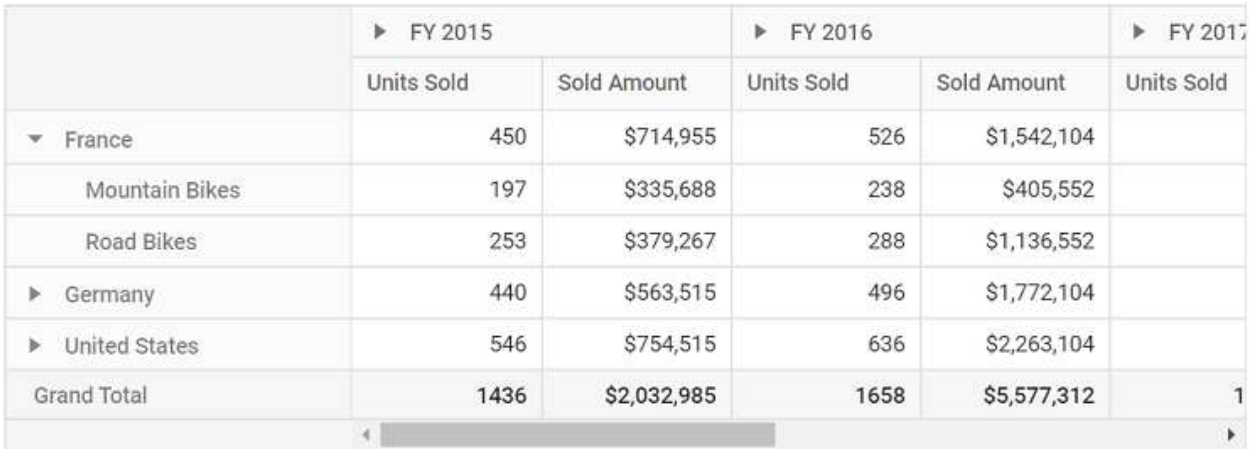
{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs204" %}

Drill down in Vue Pivotview component

Drill down and drill up

The drill down and drill up action helps to view the bound data in detailed and abstract view respectively. By default, if member(s) has children, then expand and collapse icon will be displayed in the respective row/column header. On clicking the icon, expand or collapse action will be performed automatically through built-in source code. Meanwhile, leaf member(s) does not contain expand and collapse icon.

Copyright © 2001 -2024 Syncfusion Inc.

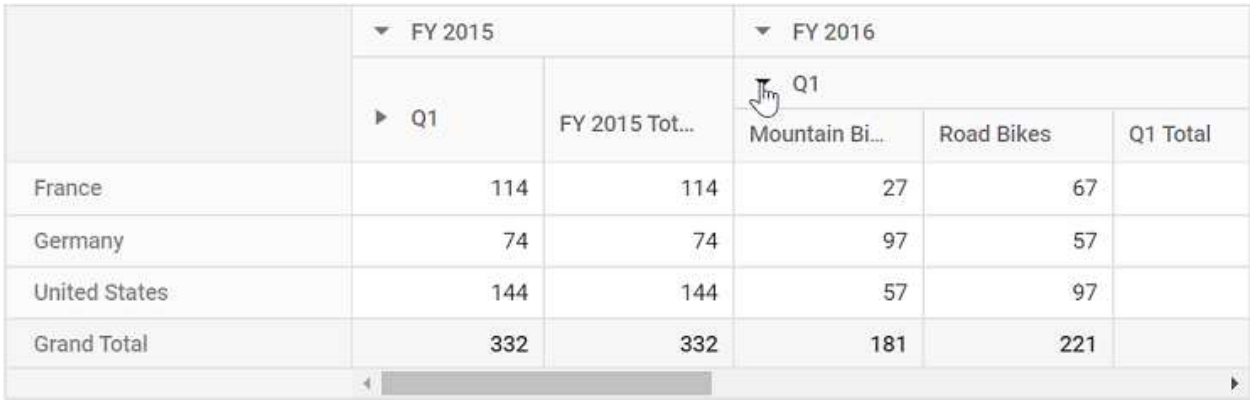


	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
France	450	$714,955	526	$1,542,104	
Mountain Bikes	197	$335,688	238	$405,552	
Road Bikes	253	$379,267	288	$1,136,552	
Germany	440	$563,515	496	$1,772,104	
United States	546	$754,515	636	$2,263,104	
Grand Total	1436	$2,032,985	1658	$5,577,312	1

Drill position

Allows to drill only the current position of the selected member and exclude the drilled data of selected member in other positions. For example, if "FY 2015" and "FY 2016" have "Quarter 1" member as child in next level, and when end user attempts to drill "Quarter 1" under "FY 2016", only it will be expanded and not "Quarter 1" under "FY 2015".

This feature is built-in and occurs every time when expand or collapse action is done for better performance.



	FY 2015		FY 2016		
	Q1	FY 2015 Tot...	Q1		
			Mountain Bi...	Road Bikes	Q1 Total
France	114	114	27	67	
Germany	74	74	97	57	
United States	144	144	57	97	
Grand Total	332	332	181	221	

Expand all

This property is applicable only for the relational data source.

Allows to either expand or collapse all headers that are displayed in row and column axes. To display all headers in expanded state, set the property expandAll to **true** and to collapse all headers, set the property expandAll to **false**. By default, expandAll property is set to **false**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs27" %}

Expand all headers for specific fields

This property is applicable only for the relational data source.

Allows to expand or collapse all headers for specific fields (only) in row and column axes. To expand headers for a specific field in row or column axis, set the property expandAll in rows or columns to **true**. By default, expandAll property in rows or columns is set to **false**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year', expandAll:
true }, { name: 'Quarter' }],
```

```
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country', expandAll: true }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs28" %}

Expand all except specific member(s)

This option is applicable only for the relational data source.

In addition to the previous topic, there is an enhancement to expand all headers expect specific header(s) and similarly to collapse all headers except specific header(s). To achieve this, drilledMember is used. The required properties of the drilledMember are explained below:

- name: It allows to set the field name whose member(s) needs to be specifically drilled.
- items: It allows to set the exact member(s) which needs to be drilled.

The drilledMember option always works in vice-versa with respect to the property expandAll in pivot table. For example, if expandAll is set to **true**, then the member(s) added in items collection alone will be in collapsed state.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
```

```
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs29" %}

Expand specific member(s)

End user can also manually expand or collapse specific member(s) in each fields under row and column axes using the drilledMembers from code behind. The required properties of the drilledMembers are explained below:

- name: It allows to set the field name whose member(s) needs to be specifically drilled.
- items: It allows to set the exact member(s) which needs to be drilled.
- delimiter: It allows to separate next level of member from its parent member.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350
    }
  }
}
```

```
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs30" %}

Event

The event drill triggers every time when a field is expanded or collapsed. For instance using this event user can alter delimiter and drill action for the respective item. It has the following parameters:

- drillInfo - It holds the current drilled item information.
- pivotview - It holds pivot table instance.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, DrillArgs } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350
    }
  }
  methods: {
    drill:function(args: DrillArgs) {
        //args.drillInfo --> in this user can get which field is drilled
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs31" %}

ActionBegin

The event actionBegin triggers when the UI actions such as drill down and drill up begin. This allows user to identify the current action being performed at runtime. It has the following parameters:

- dataSourceSettings: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- actionName: It holds the name of the current action began. The following are the UI actions and their names:

| Action | Action Name|

|------|------------|

| Expand| Drill down|

| Collapse| Drill up|

- cancel: It allows user to restrict the current action.

In the below sample, drill down and drill up action can be restricted by setting the **args.cancel** option to **true** in the actionBegin event.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings" :actionBegin="actionBegin"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotActionBeginEventArgs } from "@syncfusion/ej2-
vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
```

```
      height: 350
    }
  },
  methods: {
    actionBegin: function (args: PivotActionBeginEventArgs) {
      if (args.actionName == 'Drill down' || args.actionName == 'Drill up')
{
            args.cancel = true;
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs32" %}

ActionComplete

The event actionComplete triggers when a UI action such as drill down or drill up, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- dataSourceSettings: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- actionName: It holds the name of the current action completed. The following are the UI actions and their names:

| Action | Action Name|

|------|----------|

| Expand| Drill down|

| Collapse| Drill up|

- actionInfo: It holds the unique information about the current UI action. For example, if drill down action is completed, the event argument contains information such as field name and the drill information.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings" :actionComplete="actionComplete">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotActionCompleteEventArgs } from
"@syncfusion/ej2-vue-pivotview";
```

```
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350
    }
  },
   methods: {
    actionComplete: function (args: PivotActionCompleteEventArgs) {
      if (args.actionName == 'Drill down' || args.actionName == 'Drill up')
{
            // Triggers when the drill operations are completed.
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs33" %}

ActionFailure

The event actionFailure triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- actionName: It holds the name of the current action failed. The following are the UI actions and their names:

| Action | Action Name|

|------|------------|

| Expand| Drill down|

| Collapse| Drill up|

- errorInfo: It holds the error information of the current UI action.

APP.VUE

```
<template>
```

```
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings" :actionFailure="actionFailure">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotActionFailureEventArgs } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350
    }
  },
   methods: {
    actionFailure: function (args: PivotActionFailureEventArgs) {
      if (args.actionName == 'Drill down' || args.actionName == 'Drill up')
{
        // Triggers when the current UI action fails to achieve the desired
result.
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs34" %}

Data Shaping

Aggregation in Vue Pivotview component

This feature is applicable only for the relational data source.

End user can perform calculations over a group of values (exclusively for value fields bound in value axis) using the aggregation option. By default, values are added (summed) together. The other aggregation types are explained below.

Copyright © 2001 -2024 Syncfusion Inc.

The fields with data type such as number support all aggregation types mentioned below except for **“CalculatedField”**. The fields with data type such as string, date, datetime, boolean, etc., support **“Count”** and **“DistinctCount”** aggregation types alone.

| Operator | Description |

|------|-------------|

| Sum| Displays the pivot table values with sum.|

| Product| Displays the pivot table values with product.|

| Count| Displays the pivot table values with count.|

| DistinctCount| Displays the pivot table values with distinct count.|

| Min| Displays the pivot table with minimum value.|

| Max| Displays the pivot table with maximum value.|

| Avg| Displays the pivot table values with average.|

| Index| Displays the pivot table values with index.|

| PopulationStDev| Displays the pivot table values with standard deviation of population.|

| SampleStDev| Displays the pivot table values with sample standard deviation.|

| PopulationVar| Displays the pivot table values with variance of population.|

| SampleVar| Displays the pivot table values with sample variance.|

| RunningTotals| Displays the pivot table values with running totals.|

| DifferenceFrom| Displays the pivot table values with difference from the value of the base item in the base field.|

| PercentageOfDifferenceFrom| Displays the pivot table values with percentage difference from the value of the base item in the base field.|

| PercentageOfGrandTotal| Displays the pivot table values with percentage of grand total of all values.|

| PercentageOfColumnTotal| Displays the pivot table values in each column with percentage of total values for the column.|

| PercentageOfRowTotal| Displays the pivot table values in each row with percentage of total values for the row.|

| PercentageOfParentTotal| Displays the pivot table values with percentage of total of all values based on selected field.|

| PercentageOfParentColumnTotal| Displays the pivot table values with percentage of its parent total in each column.|

| PercentageOfParentRowTotal| Displays the pivot table values with percentage of its parent total in each row.|

| CalculatedField| Displays the pivot table with calculated field values. It allows user to create a new calculated field alone.|

Copyright © 2001 -2024 Syncfusion Inc.

Assigning aggregation type for value fields through API

For each value field, the aggregation type can be set using the property type in values. Meanwhile, aggregation types like **DifferenceFrom** and **PercentageOfDifferenceFrom** can check for specific field of specific item using baseField and baseItem properties. Likewise, **PercentageOfParentTotal** type can for specific field using baseField property. For instance, the aggregation type **DifferenceFrom** would intake the specified field and its corresponding member as input and its value is compared across other members in the same field and also across different fields to formulate an appropriate output value.

- type: It allows to set the aggregate type of the field.
- baseField: It allows to set the specific field to aggregate the values.
- baseItem: It allows to set the specific member to aggregate the values.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' type:"DifferenceFrom"
baseField:"Country" baseItem:"France" }, { name: 'Amount', caption: 'Sold
Amount', type: 'Sum' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs1" %}

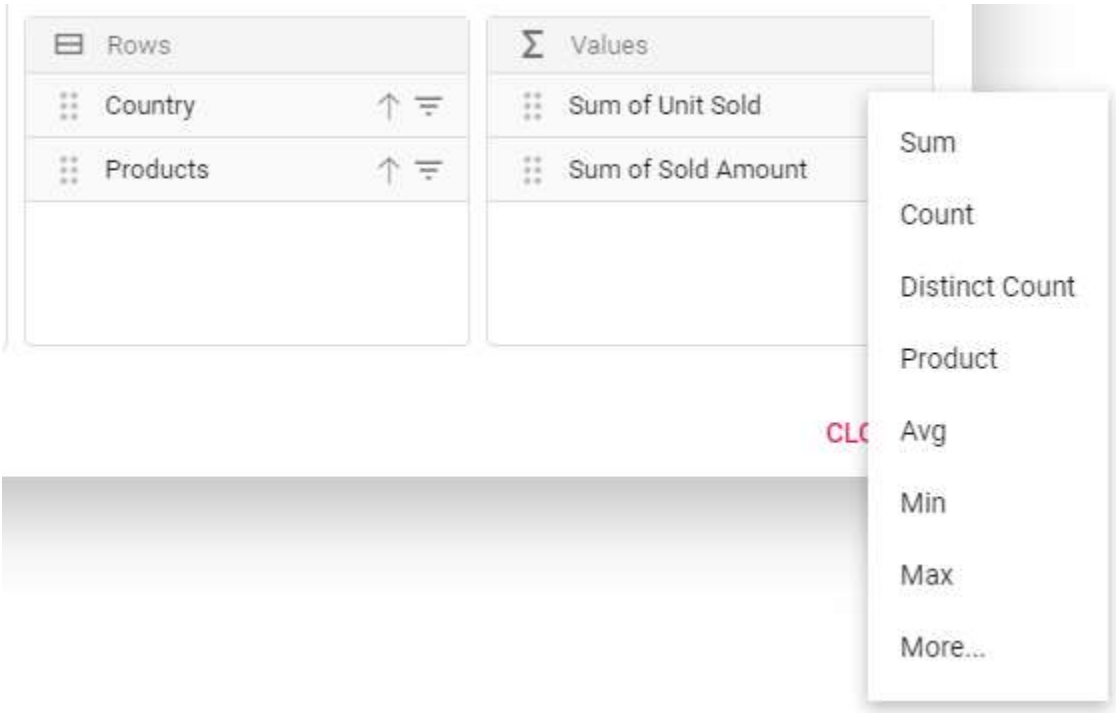
Copyright © 2001 -2024 Syncfusion Inc.

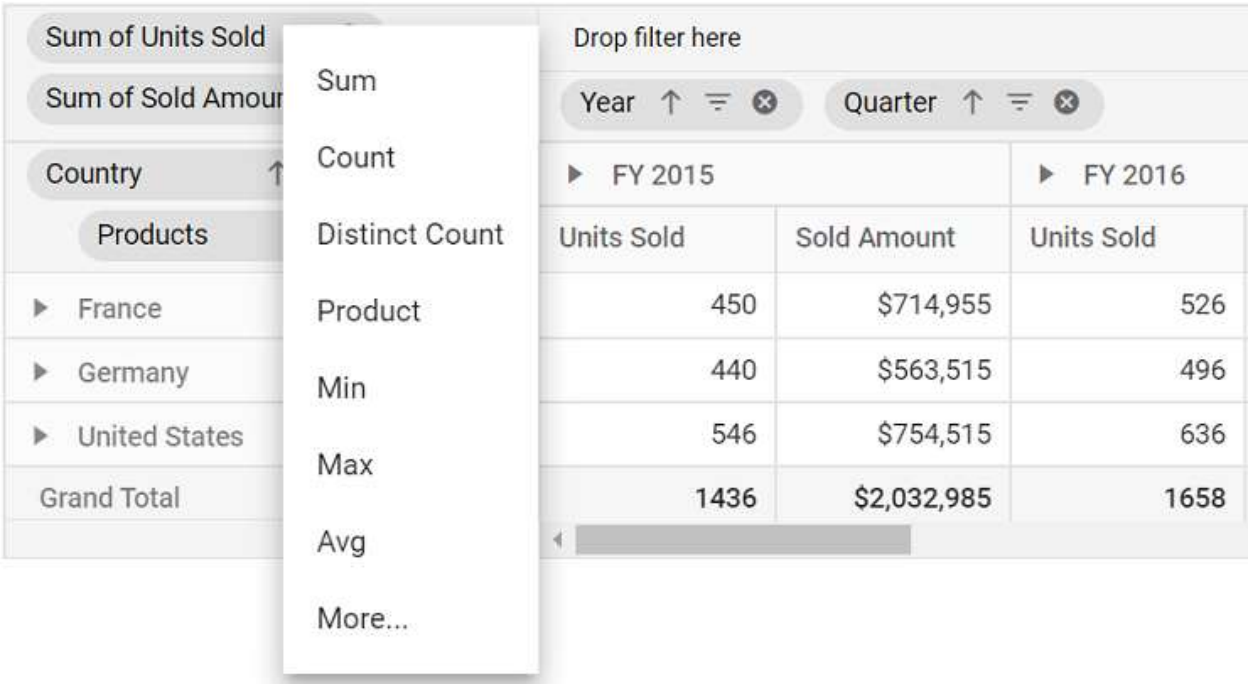
By default, the aggregation will be considered as **Sum** to the value fields which had number type and for the value fields which had non-number type values such as string, date, datetime, boolean, etc., the aggregation type will be considered as **Count**.

Modifying aggregation type for value fields at runtime

Aggregation types can be changed easily through UI at runtime. The value fields bound to grouping bar and field list appears with a dropdown icon which helps to select an appropriate aggregation type for the respective value field. On selection, the values in the pivot table will be changed dynamically.

<!-- markdownlint-disable MD012 -->





Show desired aggregation types in its dropdown menu

By default, all the aggregation types are displayed in the dropdown menu available in buttons. However, based on the request for an application, we may need to show selective aggregation types on our own. This can be achieved using the aggregateTypes property.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"
:showGroupingBar="showGroupingBar" :aggregateTypes ="aggregateTypes"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar, FieldList } from "@syncfusion/ej2-
vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount', type: 'Sum' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: [],
      },
```

```
      height: 350,
      showFieldList: true,
      showGroupingBar: true,
      aggregateTypes: ['DistinctCount', 'Avg', 'Product']
    }
  },
  provide: {
      pivotview: [GroupingBar, FieldList]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs2" %}

Hiding aggregation type from button text

By default, in value axis each field would be displayed by its name and aggregation type together. To hide aggregation type and display field name alone, set the property showAggregationOnValueField in dataSourceSettings to **false**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showGroupingBar="showGroupingBar"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar } from "@syncfusion/ej2-vue-
pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        showAggregationOnValueField: false
      },
      height: 350,
      showGroupingBar: true
    }
  },
  provide: {
```

```
        pivotview: [GroupingBar]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs3" %}

Hiding aggregation type icon from UI

By default, the icon to set aggregation type is enabled in the grouping bar. To disable this icon, set the property showValueTypeIcon in groupingBarSettings to **false**.

Icon to change the aggregation type can be hidden only in Grouping Bar but not in Field List at the moment.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showGroupingBar="showGroupingBar"
:groupingBarSettings="groupingBarSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar } from "@syncfusion/ej2-vue-
pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        showAggregationOnValueField: false
      },
      height: 350,
      showGroupingBar: true,
      groupingBarSettings: {
          showValueTypeIcon: false
      }
    }
  },
  provide: {
        pivotview: [GroupingBar]
```

```
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs4" %}

Event

AggregateCellInfo

The event aggregateCellInfo triggers every time while rendering value cell. This allows user to change the cell value and skip formatting if applied. It has following parameters:

- fieldName - It holds current cell's field name.
- row - It holds current cell's row value.
- column - It holds current cell's row value.
- value - It holds value of current cell.
- cellSets - It holds raw data for the aggregated value cell.
- rowCellType - It holds row cell type value.
- columnCellType - It holds column cell type value.
- aggregateType - It holds aggregate type of the cell.
- skipFormatting - boolean property, it allows to skip formatting if applied.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :aggregateCellInfo="aggregateCell"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
      },
      height: 350
```

```
    }
  }
  methods: {
    aggregateCell:function(args: any) {
        args.skipFormatting = false;
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs5" %}

ActionBegin

The event actionBegin triggers when clicking and selecting the aggregate type via the dropdown icon in the value field button, which is present in both grouping bar and field list UI. This allows user to identify the current action being performed at runtime. It has the following parameters:

- dataSourceSettings: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- actionName: It holds the name of the current action began. For example, while performing aggregation, the action name will be shown as **Aggregate field**.
- fieldInfo: It holds the selected value field information.

Note: This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- cancel: It allows user to restrict the current action.

In the following example, action taken during aggregation type selection via dropdown icon can be restricted by setting the **args.cancel** option to **true** in the actionBegin event.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showGroupingBar="showGroupingBar"
:actionBegin="actionBegin"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar, FieldList, PivotActionBeginEventArgs
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
```

```
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showGroupingBar: true
    }
  },
  methods: {
    actionBegin: function (args: PivotActionBeginEventArgs) {
        if (args.actionName == 'Aggregate field') {
            args.cancel = true;
        }
    }
  },
  provide: {
        pivotview: [GroupingBar, FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs6" %}

ActionComplete

The event actionComplete triggers when a UI action, such as applying aggregation using the dropdown icon via the value field button, which is present in both the grouping bar and the field list UI, is completed. This allows user to identify the current UI action being completed at runtime. It has the following parameters:

- dataSourceSettings: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- actionName: It holds the name of the current action completed. For example, after completing the aggregation, the action name will be shown as **Field aggregated**.
- fieldInfo: It holds the selected value field information.

Note: This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showGroupingBar="showGroupingBar"
```

```
:showFieldList="showFieldList" :actionComplete="actionComplete"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar, FieldList,
PivotActionCompleteEventArgs } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showGroupingBar: true,
      showFieldList: true
    }
  },
  methods: {
    actionComplete: function (args: PivotActionCompleteEventArgs) {
        if (args.actionName == 'Field aggregated') {
            // Triggers when the aggregation type is applied.
        }
    }
  },
  provide: {
        pivotview: [GroupingBar, FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs7" %}

ActionFailure

The event actionFailure triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- actionName: It holds the name of the current action failed. For example, if the action fails while performing the aggregation, then the action name will be shown as **Aggregate field**.
- errorInfo: It holds the error information of the current UI action.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showGroupingBar="showGroupingBar"
:showFieldList="showFieldList" :actionFailure="actionFailure"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar, FieldList,
PivotActionFailureEventArgs } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showGroupingBar: true,
      showFieldList: true
    }
  },
  methods: {
    actionFailure: function (args: PivotActionFailureEventArgs) {
        if (args.actionName == 'Aggregate field') {
            // Triggers when the current UI action fails to achieve the
desired result.
        }
    }
  },
  provide: {
        pivotview: [GroupingBar, FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs8" %}

Copyright © 2001 -2024 Syncfusion Inc.

Calculated field in Vue Pivotview component

Allows end user to create a new calculated field in the pivot table, based on available fields from the bound data source or using simple formula with basic arithmetic operators. It can be added at runtime through the built-in dialog, invoked from Field List UI. To do so, set the allowCalculatedField property in pivot table to **true** in the pivot table. End user can now see a "CALCULATED FIELD" button enabled in Field List UI automatically, which on clicking will invoke the calculated field dialog and perform necessary operation.

Calculated field can also be included in the pivot table through code behind using the calculatedFieldSettings. The required properties to create a new calculate field are:

- name: It allows to indicate the calculated field with a unique name.
- formula: It allows to set the formula.
- formatString: It helps to set the number format for the resultant value.

To use calculated field option, you need to inject the `CalculatedField` module in pivot table.

The calculated field is applicable only for value fields. Also, calculated field created through code behind will be automatically listed in the UI dialog as well.

APP.VUE

```
<template>
    <div id="app">
      <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :showFieldList="showFieldList"
:allowCalculatedField="allowCalculatedField"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, CalculatedField, FieldList } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }, { name: 'Total', caption: 'Total Amount', type:
'CalculatedField' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }, { name: 'Total',
format: 'C2' }],
        filters: [],
        calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Sold)"' }]
      },
```

```
      height: 350,
      allowCalculatedField: true,
      showFieldList: true,
      isPrimary: true
    }
  },
  provide: {
        pivotview: [CalculatedField, FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs9" %}

Meanwhile, you can also display the calculated field dialog independently through other means. For example, you can invoke the dialog on an external button click which is shown in the below code sample.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="calculated-field-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Calculated Field</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowCalculatedField="allowCalculatedField"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, CalculatedField } from "@syncfusion/ej2-vue-
pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }, { name: 'Total', caption: 'Total Amount', type:
'CalculatedField' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }, { name: 'Total',
format: 'C2' }],
```

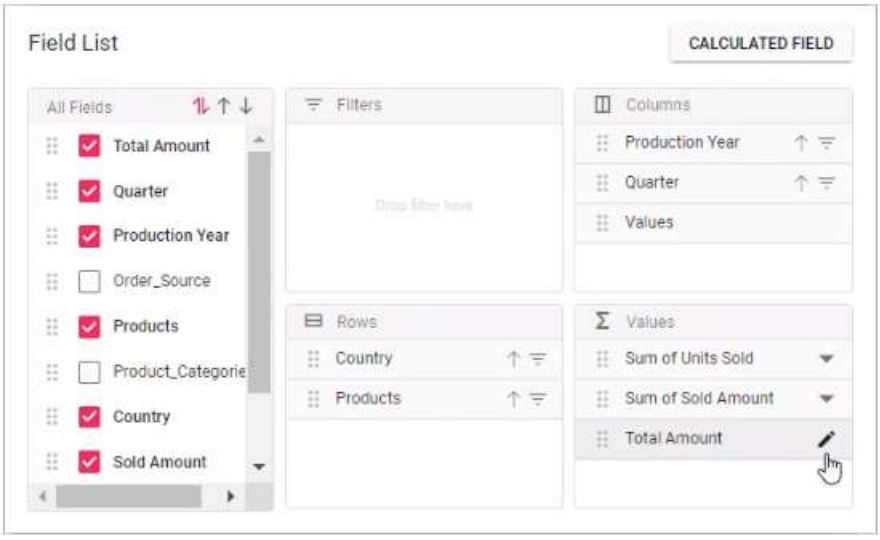
Copyright © 2001 -2024 Syncfusion Inc.

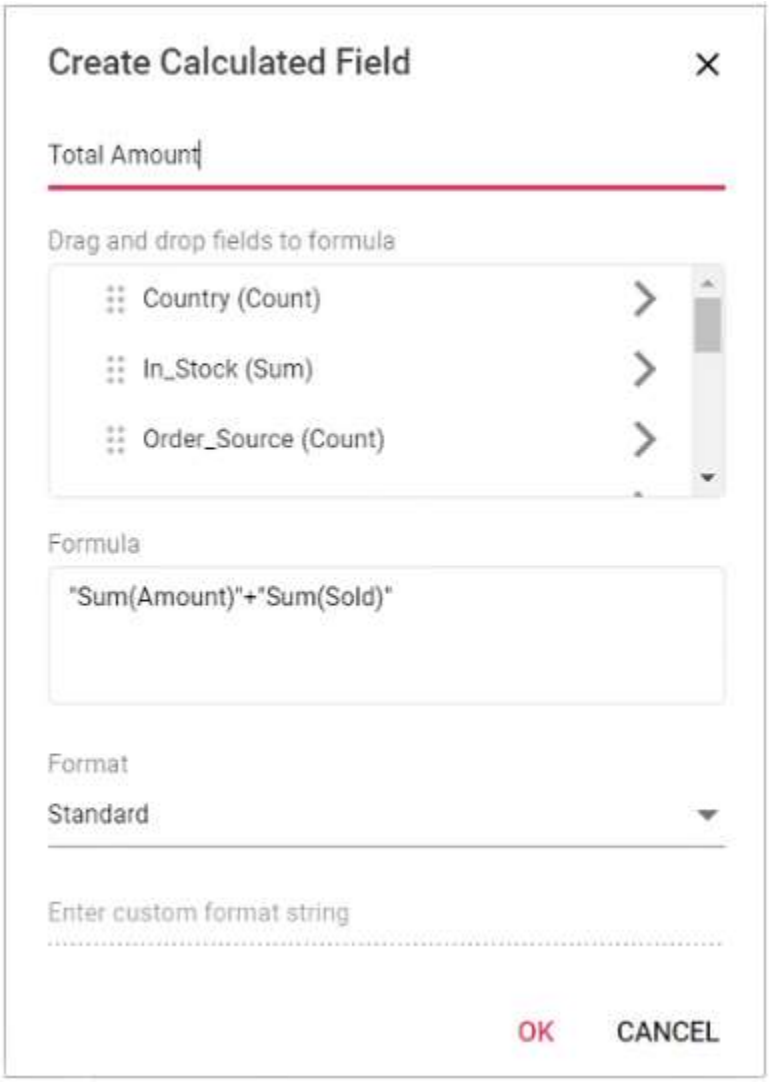
```
        filters: [],
        calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Sold)"' }]
      },
      height: 320,
      allowCalculatedField: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.calculatedFieldModule.createCalculatedFieldDialog();
    }
  },
  provide: {
        pivotview: [CalculatedField]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs10" %}

Editing through the field list and the grouping bar

User can also modify the existing calculated field using the built-in edit option available directly in the field list (or) grouping bar. To do so, click the "Edit" icon available in the calculated field button. Now the calculated field dialog is opened and the current calculated field name, formula and format can be changed at runtime.

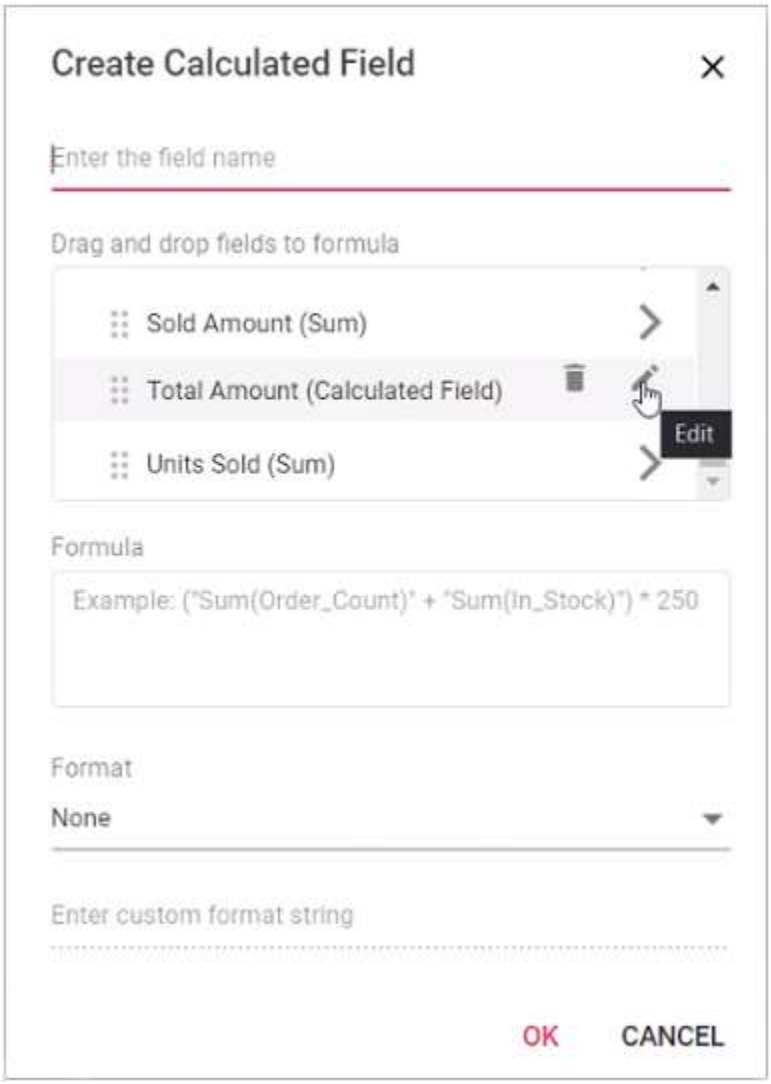


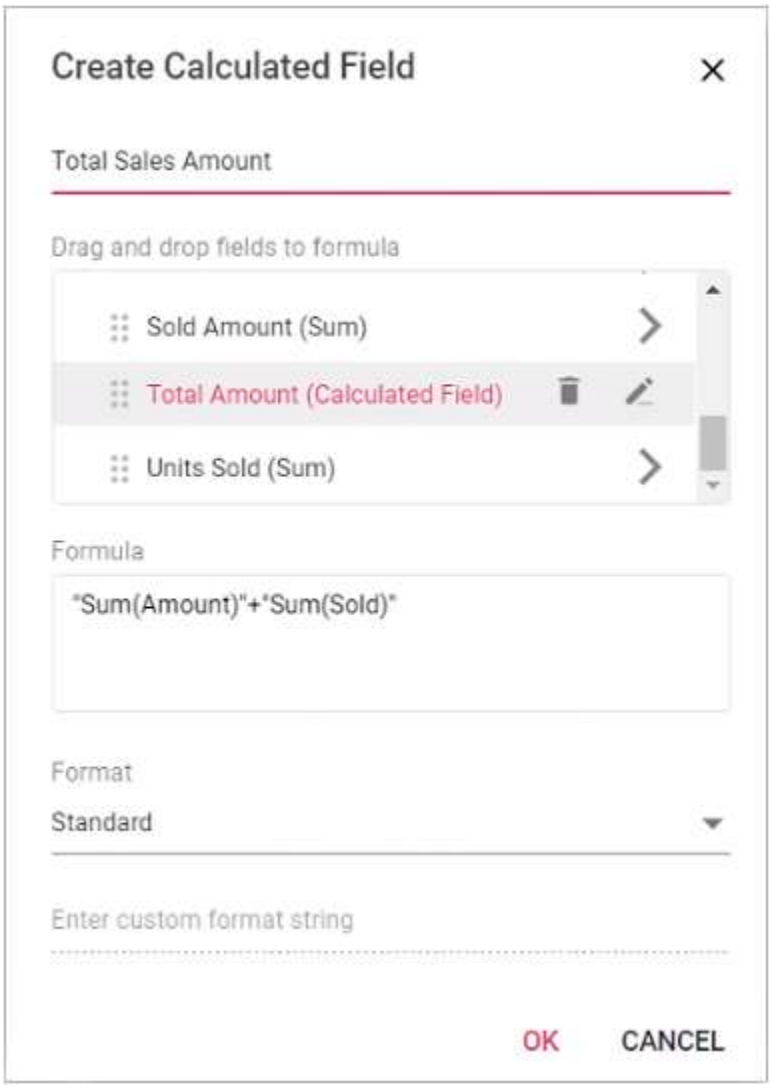


Renaming the existing calculated field

Existing calculated field can be renamed only through the UI at runtime. To do so, open the calculated field dialog, select the target field and click "Edit" icon. User can now see the existing name getting displayed in the text box at the top of the dialog. Now, change the name based on user requirement and click "OK".

<!-- markdownlint-disable MD012 -->

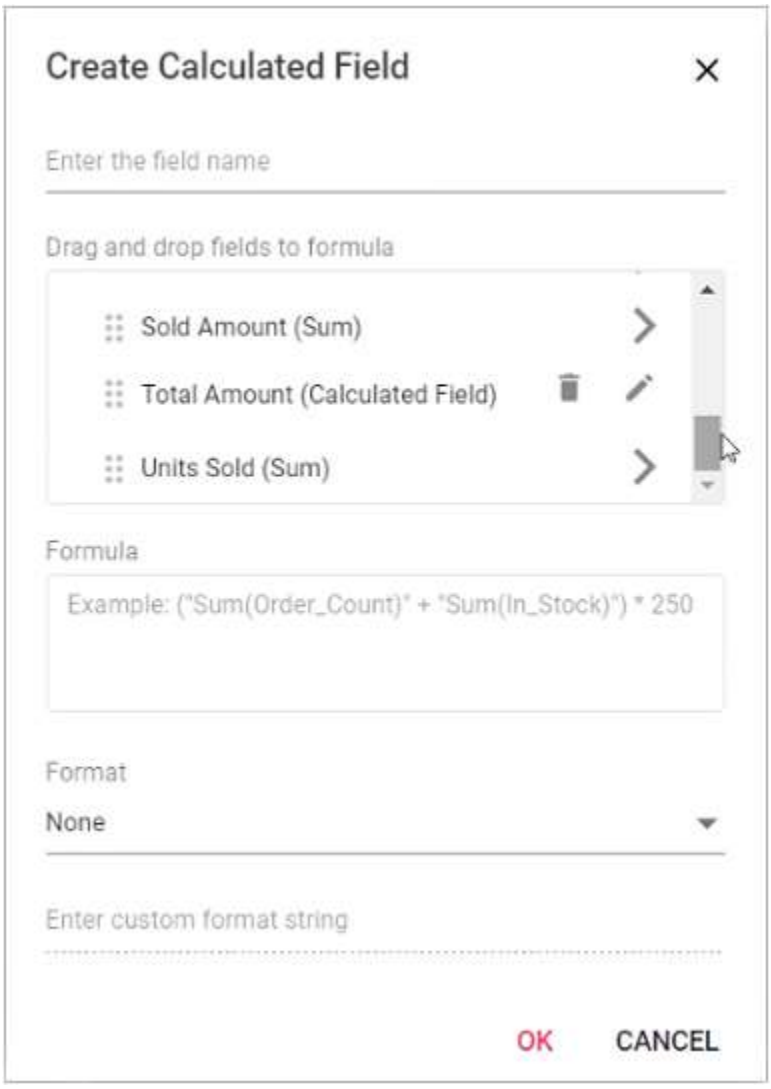

Create Calculated Field
Enter the field name
Drag and drop fields to formula
Sold Amount (Sum)
Total Amount (Calculated Field)
Edit
Units Sold (Sum)
Formula
Example: ("Sum(Order_Count)" + "Sum(In_Stock)") * 250
Format
None
Enter custom format string
OK
CANCEL

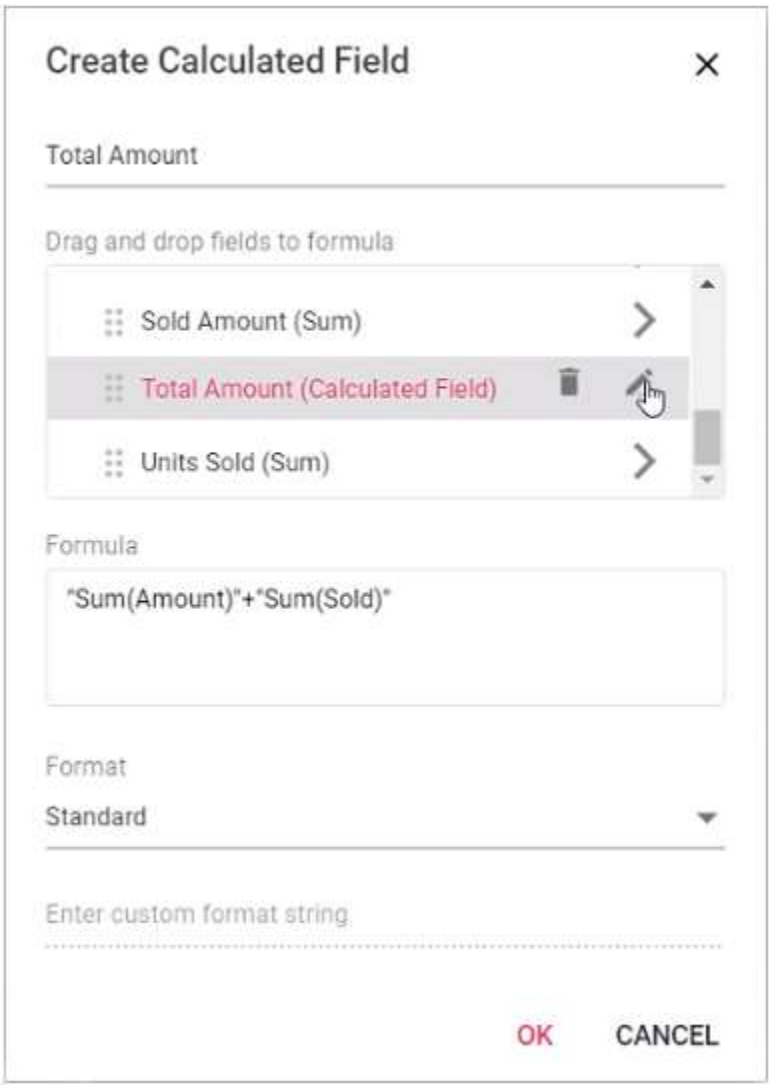


Editing the existing calculated field formula

Existing calculated field formula can be edited only through the UI at runtime. To do so, open the calculated field dialog, select the target field and click "Edit" icon. User can now see the existing formula getting displayed in a multiline text box at the bottom of the dialog. Now, change the formula based on user requirement and click "OK".

Copyright © 2001 -2024 Syncfusion Inc.

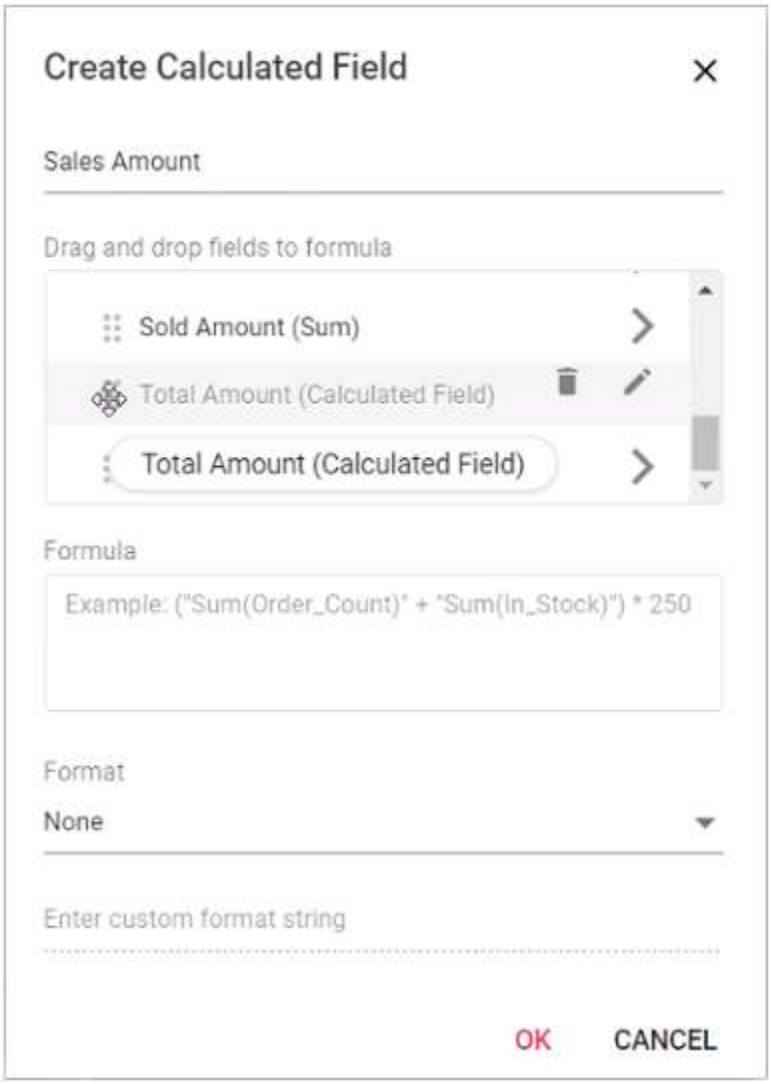

Create Calculated Field
Enter the field name
Drag and drop fields to formula
Sold Amount (Sum)
Total Amount (Calculated Field)
Units Sold (Sum)
Formula
Example: ("Sum(Order_Count)" + "Sum(In_Stock)") * 250
Format
None
Enter custom format string
OK
CANCEL

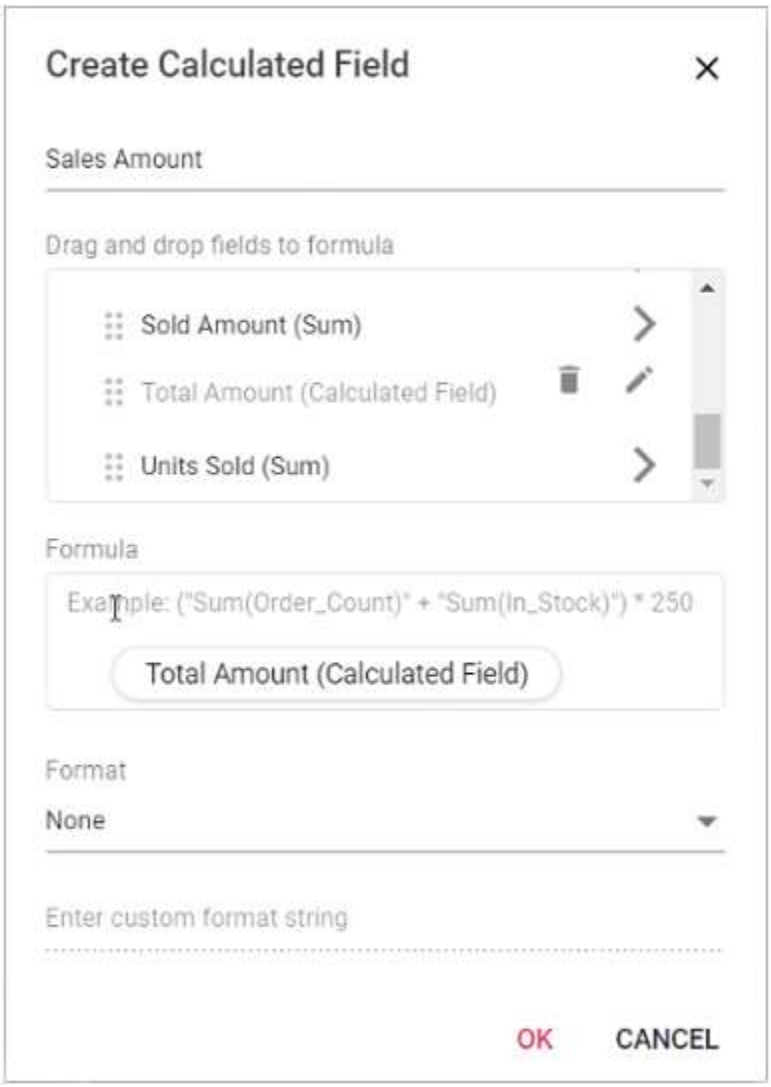


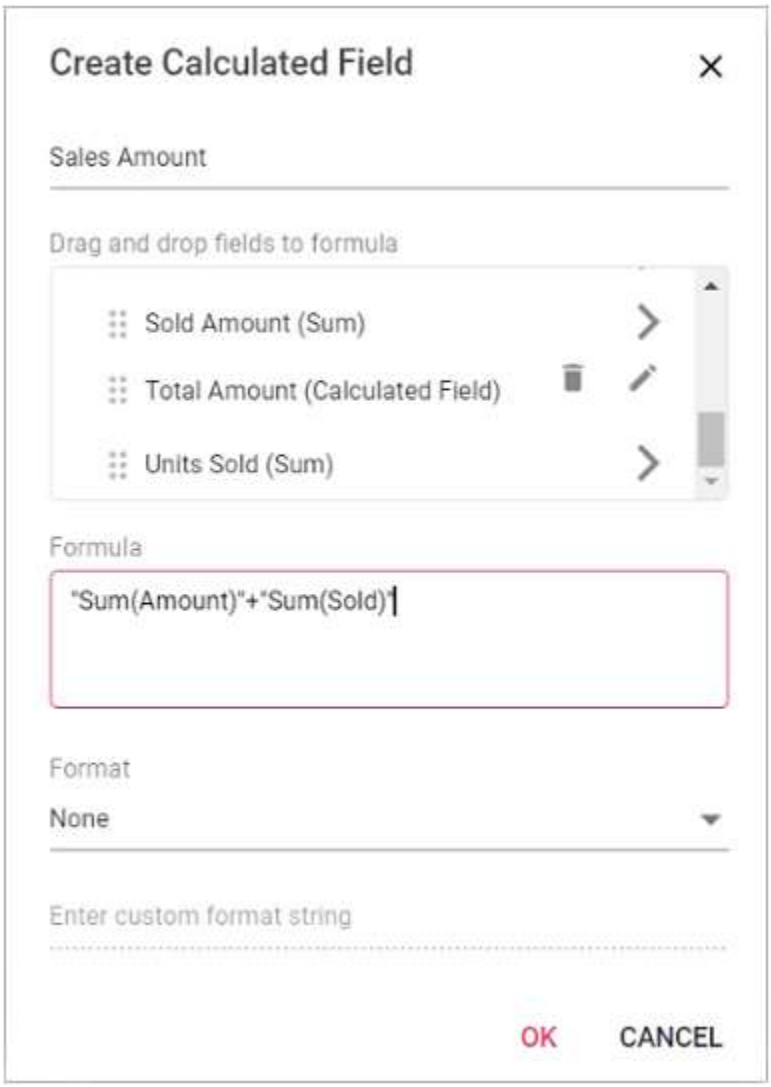
Reusing the existing formula in a new calculate field

While creating a new calculated field, if user wants to the add the formula of an existing calculated field, it can be done easily. To do so, simply drag-and-drop the existing calculated field to the "Formula" section.

Copyright © 2001 -2024 Syncfusion Inc.


Create Calculated Field
Sales Amount
Drag and drop fields to formula
Sold Amount (Sum)
Total Amount (Calculated Field)
Total Amount (Calculated Field)
Formula
Example: ("Sum(Order_Count)" + "Sum(In_Stock)") * 250
Format
None
Enter custom format string
OK
CANCEL


Create Calculated Field
Sales Amount
Drag and drop fields to formula
Sold Amount (Sum)
Total Amount (Calculated Field)
Units Sold (Sum)
Formula
Example: ("Sum(Order_Count)" + "Sum(In_Stock)") * 250
Total Amount (Calculated Field)
Format
None
Enter custom format string
OK
CANCEL



Apply the format to the calculated field values

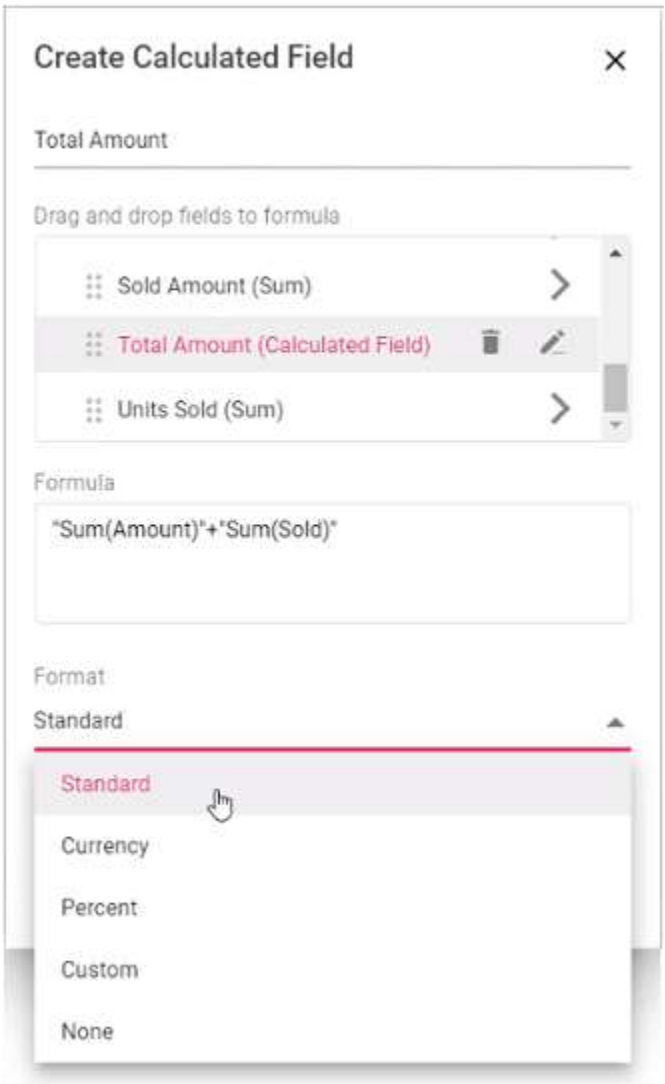
Values in a new or existing calculated field can be formatted via the calculated field UI or code behind. The formatSettings property in code-behind can be used to specify the desired format. For more information about the supported formats refer here.

To apply format to calculated field values at runtime via UI, a built-in dropdown under the "Format" label is available, from which the user can select the pre-defined format options listed below.

- **Standard** - Denotes the numeric type.
- **Currency** - Denotes the currency type.
- **Percent** - Denotes the percentage type.
- **Custom** - Denotes the custom format. For example: "C2". This shows the value "9584.3" as "$9584.30."
- **None** - Denotes that no format will be applied.

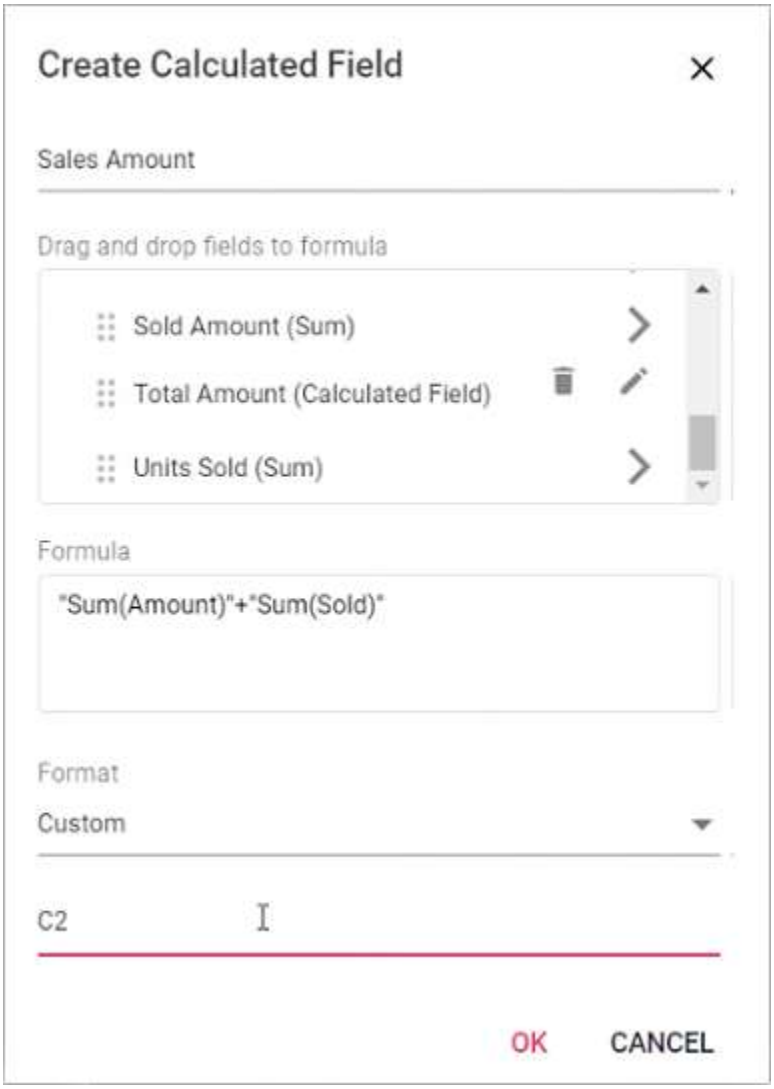
By default, **None** will be selected from the dropdown.

Copyright © 2001 -2024 Syncfusion Inc.



In addition, you can specify the desired custom formats by selecting the **Custom** option from the "Format" dropdown.

Copyright © 2001 -2024 Syncfusion Inc.



Supported operators and functions for the calculated field formula

Below is a list of operators and functions that can be used in the formula to create the calculated fields.

- + – addition operator.

`ts

Syntax: X + Y

`

- - – subtraction operator.

`ts

Syntax: X - Y

`

- * – multiplication operator.

`ts

Syntax: X * Y

`

- / – division operator.

`ts

Syntax: X / Y

`

- `^` – power operator.

Grouping in Vue Pivotview component

This feature is applicable only for the relational data source.

Grouping is the most-useful feature in pivot table and the component automatically groups date, time, number and string. For example, the date type can be formatted and displayed based on year, quarter, month, and more. Likewise, the number type can be grouped range-wise, such as 1-5, 6-10, etc. These group fields will act as individual fields and allows users to drag them between different axes such as columns, rows, values, and filters and create pivot table at runtime.

<!-- markdownlint-disable MD012 -->

Filtering in Vue Pivotview component

Filtering allows to view the pivot table with selective records based on members that can be either included or excluded through UI and code-behind.

The following are the three different types of filtering:

- Member filtering
- Label filtering
- Value filtering

When all the above filtering options are disabled via code-behind, then the filter icon would be disabled in the field list or grouping bar UI.

Member filtering

Sorting in Vue Pivotview component

Member Sorting

Allows to order field members in rows and columns either in ascending or descending order. By default, field members in rows and columns are in ascending order.

Drill through in Vue Pivotview component

Editing in Vue Pivotview component

This feature is applicable only for the relational data source.

Copyright © 2001 -2024 Syncfusion Inc.

Data Formatting

Number formatting in Vue Pivotview component

Allows you to specify the required display format that should be used in values of the pivot table. Supported display formats are:

- Number
- Currency
- Percentage
- Custom

Conditional formatting in Vue Pivotview component

Allows end user to change the appearance of the pivot table value cells with its background color, font color, font family, and font size based on specific conditions.

Report Manipulation

Grouping bar in Vue Pivotview component

Field list in Vue Pivotview component

The pivot table provides a built-in Field List similar to Microsoft Excel. It allows to add or remove fields and also rearrange them between different axes, including column, row, value, and filter along with sort and filter options dynamically at runtime.

The field list can be displayed in two different formats to interact with pivot table. They are:

- **In-built Field List (Popup)**: To display the field list icon in pivot table UI to invoke the built-in dialog.
- **Stand-alone Field List (Fixed)**: To display the field list in a static position within a web page.

In-built Field List (Popup)

To enable the field list in pivot table UI, set the showFieldList property in pivot view to **true**. A small icon will appear on the top left corner of the pivot table and clicking on this icon, field list dialog will appear.

The field list icon will be displayed at the top right corner of the pivot table, when grouping bar is enabled.

To use field list, you need to inject the FieldList module in pivot table.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
```

```
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs61" %}

Stand-alone Field List (Fixed)

The field list can be rendered in a static position, anywhere in web page layout, like a separate component. To do so, you need to set renderMode property to **Fixed** in pivot fieldlist.

To make field list interact with pivot table, you need to use the **updateView** and **update** methods for data source update in both field list and pivot table simultaneously.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview_flist" :height="height"></ejs-
pivotview>
        <ejs-pivotfieldlist id="pivotfieldlist1"
:dataSourceSettings="dataSourceSettings"
:enginePopulated="fieldEnginePopulated" :renderMode="renderMode"></ejs-
pivotfieldlist>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotFieldListPlugin, FieldList } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(PivotFieldListPlugin);
export default {
```

```
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      renderMode: "Fixed"
    }
  },
  methods: {
    fieldEnginePopulated: function(args) {
      let fieldListObj =
document.getElementById('pivotfieldlist1').ej2_instances[0];
      let pivotGridObj =
document.getElementById('pivotview_flist').ej2_instances[0];
      fieldListObj.updateView(pivotGridObj);
    }
  }
}
</script>
<style>
    @import "@syncfusion/ej2-vue-pivotview/styles/material.css";
    #pivotfieldlist1 {
      width: 400px;
      margin-top: 20px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs62" %}

Search desired field

End user can search for desired field in the field list UI by typing the field name into the search box at runtime. It can be enabled by setting the enableFieldSearching property to **true** via code-behind.

By default, field search option is disabled in the field list UI.

To enable search box in the static field list UI, set the enableFieldSearching property to **true** in ejs-pivotfieldlist.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview_flist" :height="height"></ejs-
pivotview>
        <ejs-pivotfieldlist id="pivotfieldlist1"
:dataSourceSettings="dataSourceSettings"
```

```
:enginePopulated="fieldEnginePopulated" :renderMode="renderMode"
:enableFieldSearching="enableFieldSearching"></ejs-pivotfieldlist>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotFieldListPlugin, FieldList } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(PivotFieldListPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      renderMode: "Fixed",
      enableFieldSearching: true
    }
  },
  methods: {
    fieldEnginePopulated: function(args) {
      let fieldListObj =
document.getElementById('pivotfieldlist1').ej2_instances[0];
      let pivotGridObj =
document.getElementById('pivotview_flist').ej2_instances[0];
      fieldListObj.updateView(pivotGridObj);
    }
  }
}
</script>
<style>
    @import "@syncfusion/ej2-vue-pivotview/styles/material.css";
    #pivotfieldlist1 {
      width: 400px;
      margin-top: 20px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs63" %}

To enable search box in the pivot table's built-in popup field list UI, set the enableFieldSearching property to **true** in ejs-pivotview.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"
:enableFieldSearching="enableFieldSearching"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true,
      enableFieldSearching: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

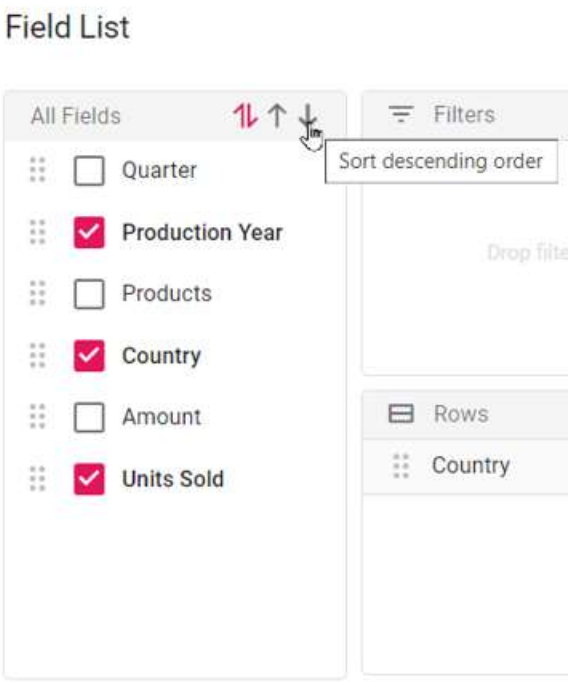
{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs64" %}

Option to sort fields

End user can sort fields in the field list UI to ascending (or) descending (or) default order (as obtained from the data source) using the built-in sort icons.

By default, fields are displayed in the default order.

Copyright © 2001 -2024 Syncfusion Inc.



Sort fields in a desired order

To display the fields in descending order by default, set the defaultFieldListOrder property to **Descending** in the load event.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showGroupingBar="showGroupingBar"
:showFieldList="showFieldList" :load="load"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar, FieldList, LoadEventArgs } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        columns: [{ name: 'Year', caption: 'Production Year' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }],
        rows: [{ name: 'Country' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        filters: [],
      },
      height: 350,
      showGroupingBar: true,
      showFieldList: true
    }
  },
  methods: {
    load: function (args: LoadEventArgs) {
      args.defaultFieldListOrder = 'Descending';
    }
  },
  provide: {
        pivotview: [GroupingBar, FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs261" %}

Group fields under desired folder name

In the field list UI, you can display fields by grouping them under the desired folder name. It can only be configured via code-behind by setting the groupName property in fieldMapping.

You can only group fields to one level using the groupName property.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"
:showGroupingBar="showGroupingBar"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList, GroupingBar } from "@syncfusion/ej2-
vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
      dataSource: pivotData,
      expandAll: false,
      allowLabelFilter: true,
      allowValueFilter: true,
      values: [{ name: 'Sold', caption: 'Units Sold' }],
      columns: [{ name: 'Year', caption: 'Production Year' }],
      rows: [{ name: 'Country' }],
      formatSettings: [{ name: 'Amount', format: 'C0' }],
```

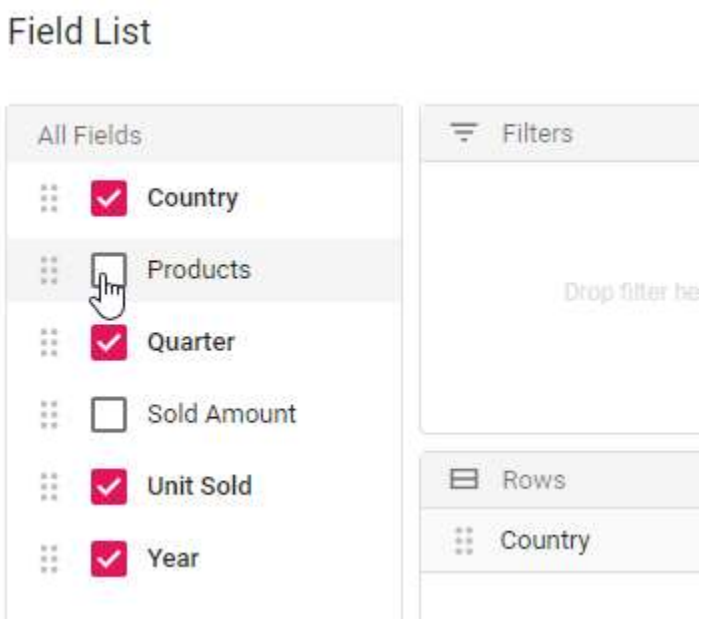
Copyright © 2001 -2024 Syncfusion Inc.

```
      filters: [],
      fieldMapping: [
        { name: 'Quarter', groupName: 'Product category' },
        { name: 'Products', groupName: 'Product category' },
        { name: 'Amount', groupName: 'Product category', caption: 'Sold
Amount' },
      ]
      },
      height: 350,
      showFieldList: true,
      showGroupingBar: true
    }
  },
  provide: {
        pivotview: [FieldList, GroupingBar]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs65" %}

Add or remove fields

Using check box besides each field, end user can select or unselect to add or remove fields respectively from the report at runtime.



Remove specific field(s) from displaying

When a data source is bound to the component, fields will be automatically populated inside the Field List. In such case, user can also restrict specific field(s) from displaying. To do so, set the appropriate field name(s) in excludeFields property belonging to dataSourceSettings.

APP.VUE

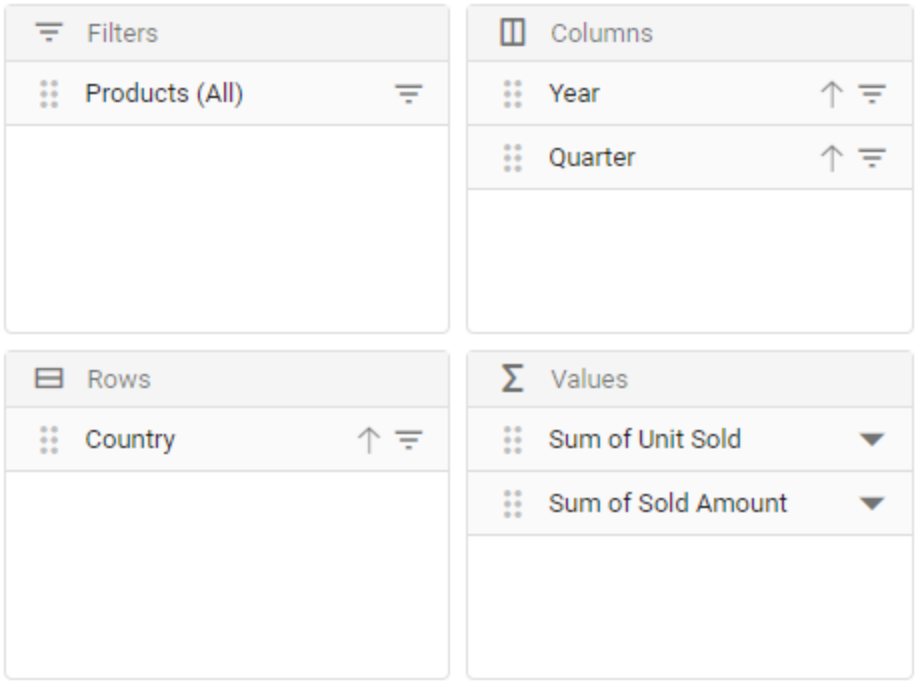
```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }],
        rows: [{ name: 'Country' }],
        excludeFields:["Products", "Amount" ],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs66" %}

Re-arranging fields

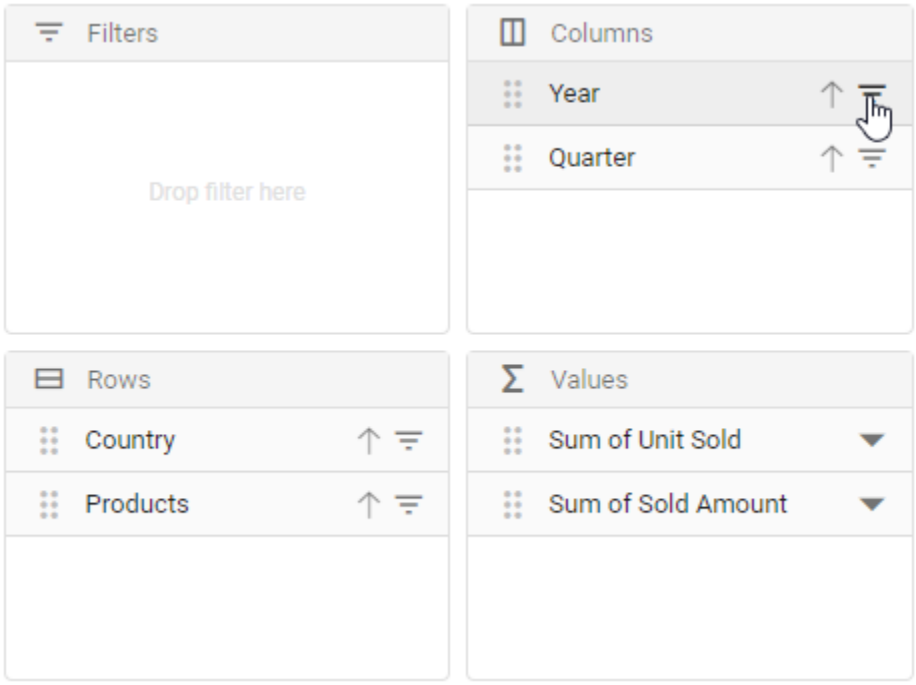
In-order to re-arrange, drag any field from the field list and drop it into the column, row, value, or filter axis using the drag-and-drop holder. It helps end user to alter the report at runtime.

Copyright © 2001 -2024 Syncfusion Inc.

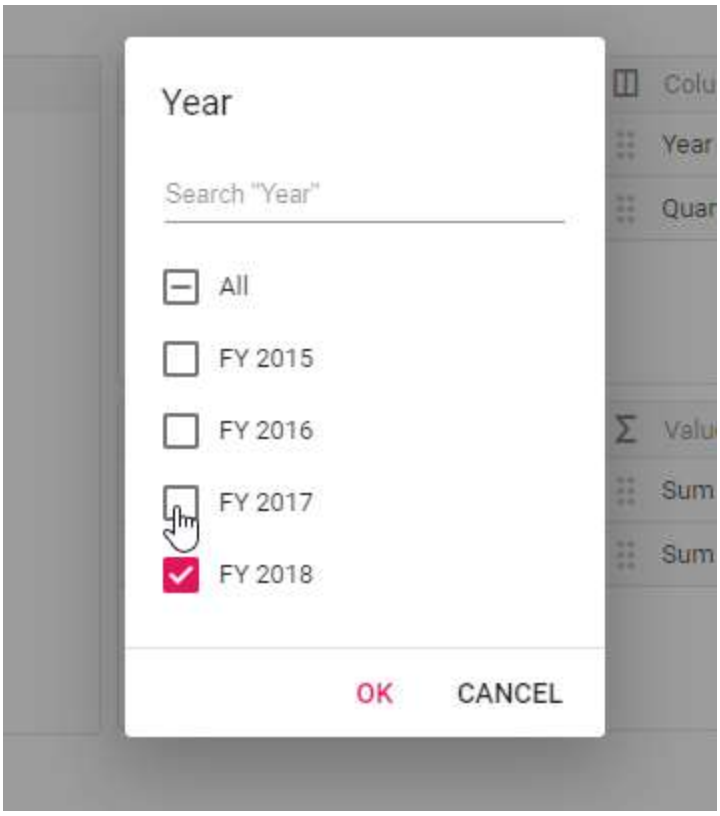


Filtering members

Using the filter icon besides each field in row, column and filter axes, members can be either included or excluded at runtime. To know more about member filtering, refer here.



Copyright © 2001 -2024 Syncfusion Inc.

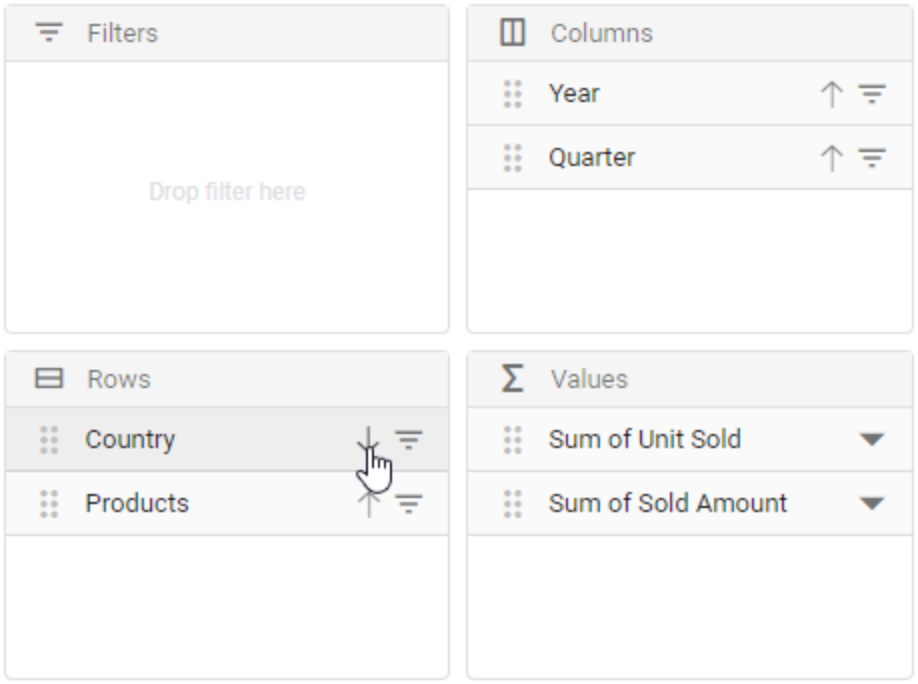


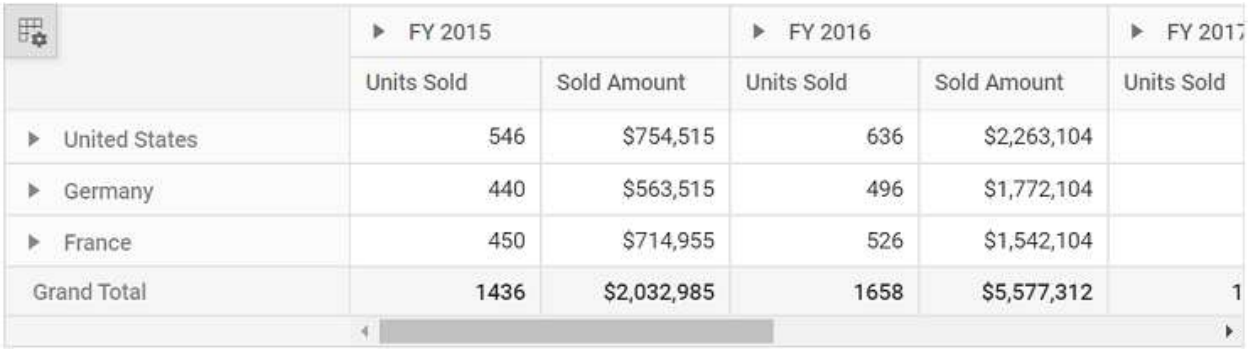
	FY 2018		Grand Total	
	Units Sold	Sold Amount	Units Sold	Sold Amount
France	16	$27,264	16	$27,264
Germany	96	$77,264	96	$77,264
United States	76	$97,264	76	$97,264
Grand Total	188	$201,792	188	$201,792

Sorting members

Using the sort icon besides each field in row and column axes, members can be arranged either in ascending or descending order at runtime. To know more about member sorting, refer here.

Copyright © 2001 -2024 Syncfusion Inc.



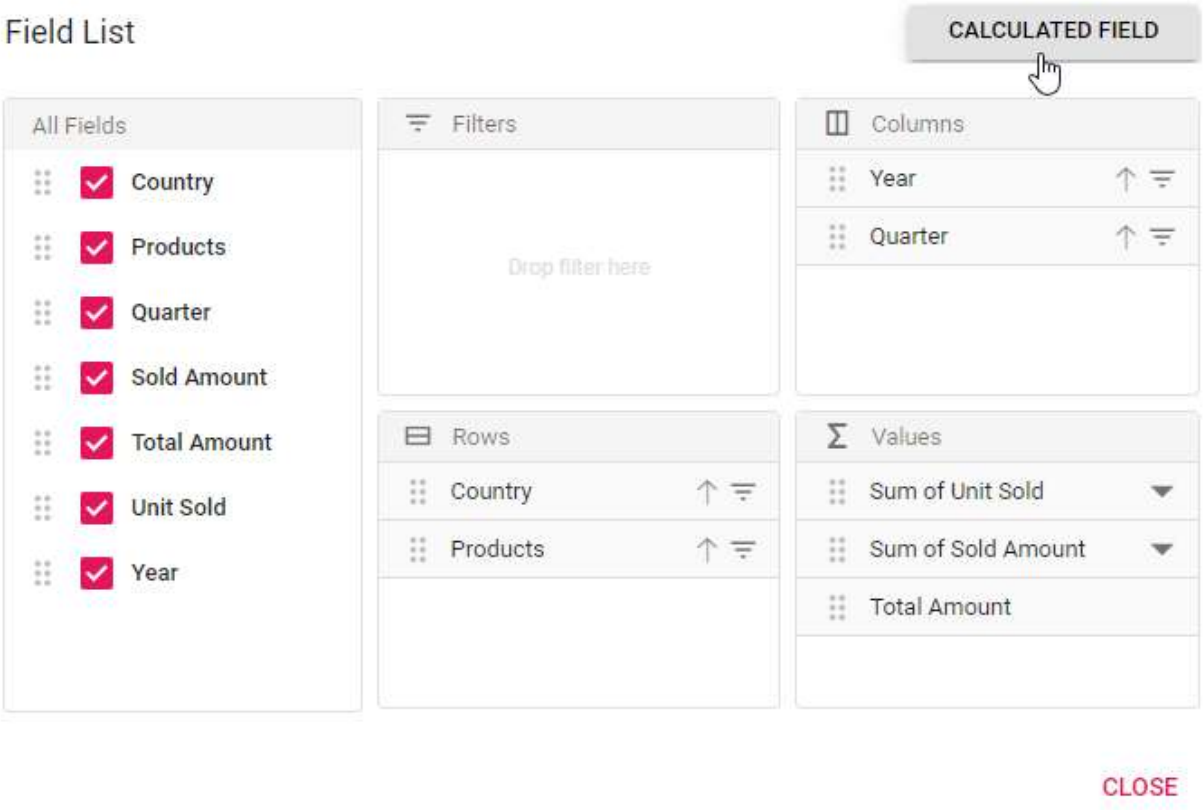


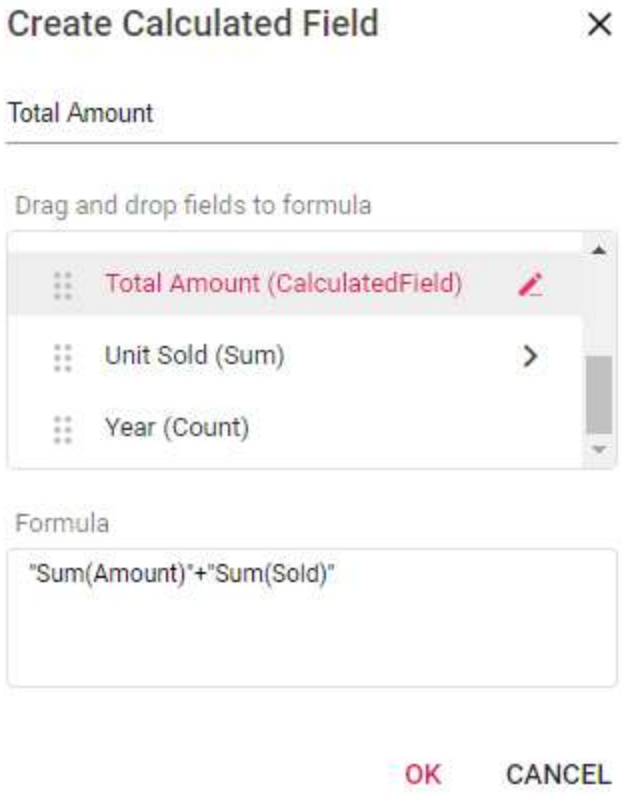
	FY 2015		FY 2016		FY 2017
	Units Sold	Sold Amount	Units Sold	Sold Amount	Units Sold
United States	546	$754,515	636	$2,263,104	
Germany	440	$563,515	496	$1,772,104	
France	450	$714,955	526	$1,542,104	
Grand Total	1436	$2,032,985	1658	$5,577,312	1

Calculated fields

The calculated field support allows end user to add a new calculated field based on the available fields from the bound data source using basic arithmetic operators. To enable this support in Field List UI, set the allowCalculatedField property in pivot table to **true** in pivot table. Now a button will be seen automatically inside the field list UI which will invoke the calculated field dialog on click. To know more about calculated field, refer here.

Copyright © 2001 -2024 Syncfusion Inc.


Field List
CALCULATED FIELD
All Fields
Country
Products
Quarter
Sold Amount
Total Amount
Unit Sold
Year
Filters
Drop filter here
Columns
Year
Quarter
Rows
Country
Products
Values
Sum of Unit Sold
Sum of Sold Amount
Total Amount
CLOSE

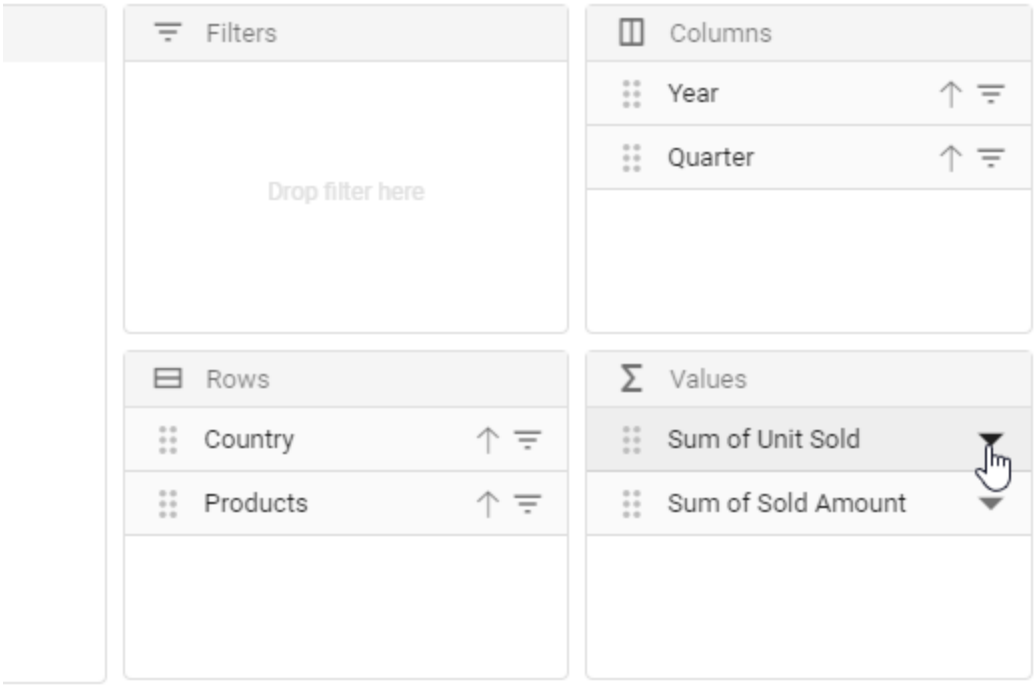


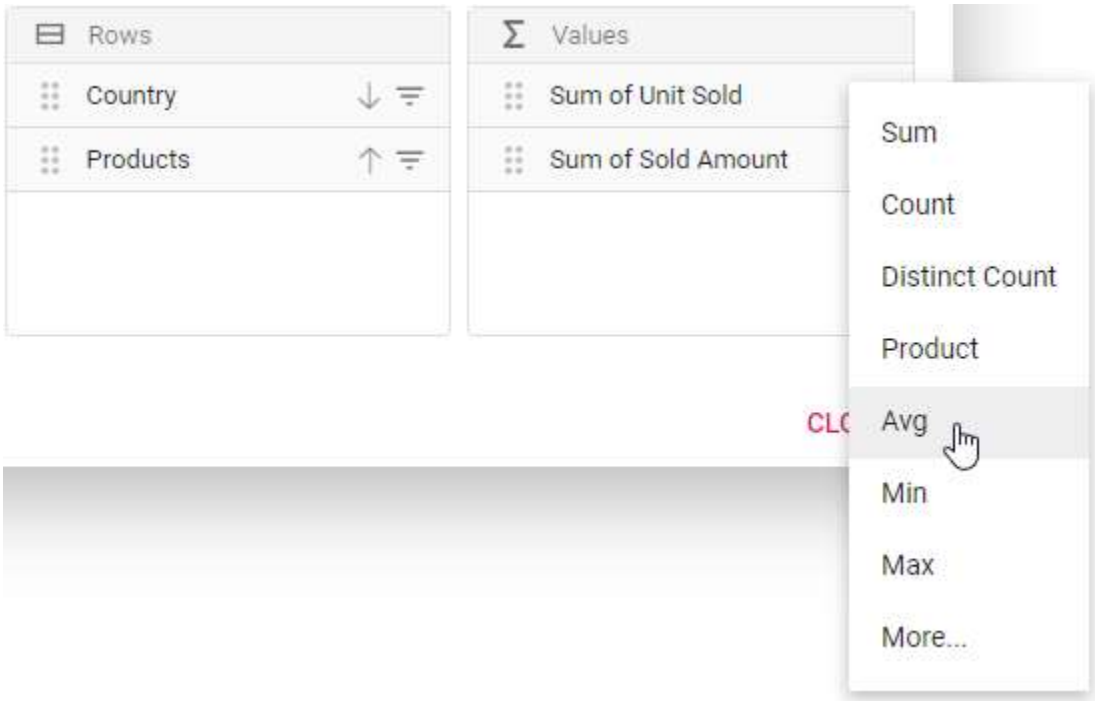
	FY 2015			FY 2016	
	Units Sold	Sold Amount	Total Amount	Units Sold	Sold Amoun
France	450	$714,955	$715,405	526	$1,542,
Germany	440	$563,515	$563,955	496	$1,772,
United States	546	$754,515	$755,061	636	$2,263,
Grand Total	1436	$2,032,985	$2,034,421	1658	$5,577,

Changing aggregation type of value fields at runtime

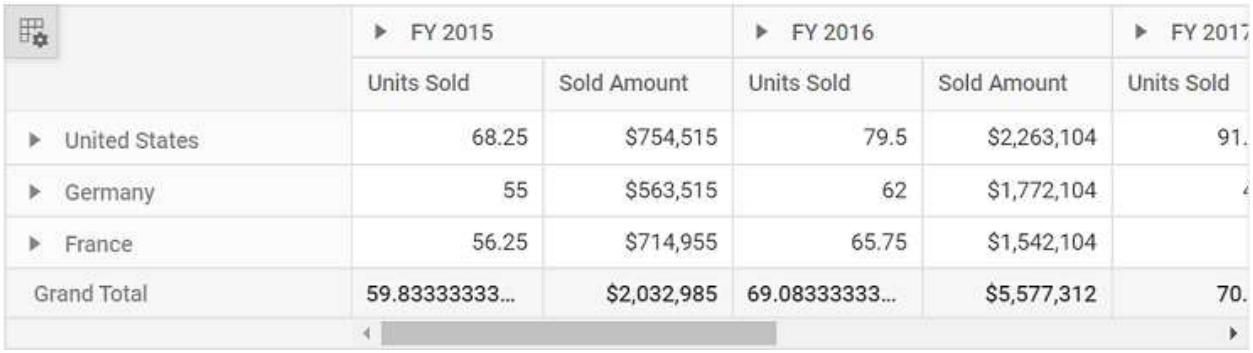
End user can perform calculations over a group of values using the aggregation option. The value fields bound to the field list, appears with a dropdown icon, helps to select an appropriate aggregation type at runtime. On selection, the values in the Pivot Table will be changed dynamically. To know more about aggregation, refer here.

Copyright © 2001 -2024 Syncfusion Inc.



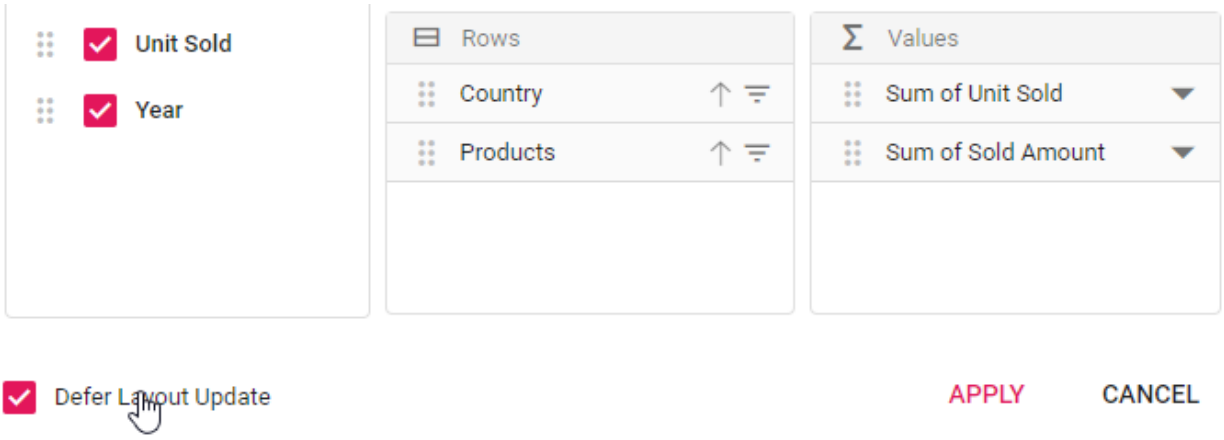


Copyright © 2001 -2024 Syncfusion Inc.



Defer layout update

Defer layout update support to update the pivot table only on demand and not during every user action. To enable this support in Field List UI, set the allowDeferLayoutUpdate property to **true** in pivot table. Now a check box inside Field List UI will be seen in checked state, allowing pivot table to update only on demand. To know more about defer layout, refer here.



Show built-in Field List (Popup) over specific target

By passing the target element to the built-in field list dialog module in the dataBound event, the field list dialog will be displayed over the appropriate target element on a web page. By default, the Pivot Table's parent element is used as the target element to display the built-in field list dialog.

The sample code below shows the built-in field list dialog using `document.body` as the target element.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList" :dataBound="ondataBound">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
```

```
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
  methods: {
    ondataBound: function() {
     let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
     pivotGridObj.pivotFieldListModule.dialogRenderer.fieldListDialog.target
= document.body;
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs67" %}

Show field list using toolbar

It can also be viewed in toolbar by setting showFieldList and showToolbar properties in pivot table to **true**. Also, include the item **FieldList** within the toolbar property in pivot table. When toolbar is enabled, field list icon will be automatically added into the toolbar and the icon won't appear on top left corner in the pivot table component.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:height="height" :showToolbar="showToolbar" :toolbar="toolbar"
:showFieldList="showFieldList" > </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import {
  PivotViewPlugin,
  IDataSet,
  Toolbar,
  FieldList
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: { columnWidth: 140 },
      showToolbar: true,
      showFieldList: true,
      toolbar: [
        "FieldList"
      ],
    };
  }
  provide: {
    pivotview: [
      Toolbar,
      FieldList
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs68" %}

Invoking dynamic Field List (Customized)

Also, you can display the field list dialog independently through other means. For example, you can invoke the field list dialog on an external button click. To do so, set renderMode property to Popup and since on button click, field list dialog will be invoked.

* Meanwhile, you can display the field list dialog over specific target element within a webpage using target property. By default, the target value is "null", which by default refers the document.body element.

Copyright © 2001 -2024 Syncfusion Inc.

* To make field list interact with pivot table, you need to use the **updateView** and **update** methods for data source update in both field list and pivot table simultaneously.

The below sample code illustrates the field list dialog invoked on an external button click.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="flist-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Show Field List</ejs-button>
        <ejs-pivotview id="pivotview_flist" :height="height"
:enginePopulated="enginePopulated"></ejs-pivotview>
        <ejs-pivotfieldlist id="pivotfieldlist1"
:dataSourceSettings="dataSourceSettings"
:enginePopulated="fieldEnginePopulated" :renderMode="renderMode"
:target="target"></ejs-pivotfieldlist>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotFieldListPlugin, FieldList } from
"@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(PivotFieldListPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      renderMode: 'Popup',
      height: 320,
      target: '#pivotfieldlist1',
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let fieldListObj =
document.getElementById('pivotfieldlist1').ej2_instances[0];
      fieldListObj.dialogRenderer.fieldListDialog.show();
    },
    fieldEnginePopulated: function() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      let fieldListObj =
document.getElementById('pivotfieldlist1').ej2_instances[0];
      let pivotGridObj =
document.getElementById('pivotview_flist').ej2_instances[0];
      fieldListObj.updateView(pivotGridObj);
    },
    enginePopulated: function() {
      let fieldListObj =
document.getElementById('pivotfieldlist1').ej2_instances[0];
      if(fieldListObj) {
        fieldListObj.updateView(pivotGridObj);
      }
    }
  }
}
</script>
<style>
    @import "@syncfusion/ej2-vue-pivotview/styles/material.css";
    .e-toggle-field-list {
      display: none !important;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs69" %}

Set caption to fields which isn't bound to the report

One can set the caption to all fields from the data source even if it is not bound to the actual report. It can be achieved using the enginePopulated event. On doing so, caption of the respective field will be displayed in both grouping bar and field list.

In the sample, we have set caption to the fields Year and Quarter dynamically.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview  id="pivotview_flist"
:dataSourceSettings="dataSourceSettings" :height="height"
:showFieldList="showFieldList" :enginePopulated="fieldEnginePopulated">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Products' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
```

```
        rows: [{ name: 'Country' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    },
  methods: {
    fieldEnginePopulated: function(args) {
      let pivotTableObj =
document.getElementById('pivotview_flist').ej2_instances[0];
      Object.keys(pivotTableObj.engineModule.fieldList).forEach((key, index)
=> {
                if (key === 'Quarter') {
                    pivotTableObj.engineModule.fieldList[key].caption =
'Production Quarter Year';
                }
                else if (key === 'Year') {
                    pivotTableObj.engineModule.fieldList[key].caption =
'Production Year';
                }
      });
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs70" %}

Show values button

During runtime, the **Values** button in the field list can be moved to a different position (i.e., different index) among other fields in the column or row axis. To enable the **Values** button, set the showValuesButton property to **true**.

This support is only available for relational data sources.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview_flist" :height="height"></ejs-
pivotview>
        <ejs-pivotfieldlist id="pivotfieldlist1"
:dataSourceSettings="dataSourceSettings"
:enginePopulated="fieldEnginePopulated" :renderMode="renderMode"></ejs-
pivotfieldlist>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { PivotViewPlugin, PivotFieldListPlugin, FieldList } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(PivotFieldListPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      renderMode: "Fixed",
      showValuesButton: true,
    }
  },
  methods: {
    fieldEnginePopulated: function(args) {
      let fieldListObj =
document.getElementById('pivotfieldlist1').ej2_instances[0];
      let pivotGridObj =
document.getElementById('pivotview_flist').ej2_instances[0];
      fieldListObj.updateView(pivotGridObj);
    }
  }
}
</script>
<style>
    @import "@syncfusion/ej2-vue-pivotview/styles/material.css";
    #pivotfieldlist1 {
      width: 400px;
      margin-top: 20px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs71" %}

Events

EnginePopulated

The enginePopulated event is available in both Pivot Table and Field List.

- The event enginePopulated is triggered in field list whenever the report gets modified. The updated report is passed to the pivot table via updateView method written within this event to refresh the same.

Copyright © 2001 -2024 Syncfusion Inc.

- Likewise, enginePopulated event is triggered in pivot table whenever the report gets modified. The updated report is passed to the field list via update method written within this event to refresh the same.

The event enginePopulated is triggered after engine is populated. It has following parameters - dataSourceSettings, pivotFieldList and pivotValues.

Note: This event is not required for Popup field list since it is a in built one.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview_flist" :height="height"></ejs-
pivotview>
        <ejs-pivotfieldlist id="pivotfieldlist1"
:dataSourceSettings="dataSourceSettings"
:enginePopulated="fieldEnginePopulated" :renderMode="renderMode"></ejs-
pivotfieldlist>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotFieldListPlugin, FieldList } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(PivotFieldListPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      renderMode: "Fixed"
    }
  },
  methods: {
    fieldEnginePopulated: function(args) {
      let fieldListObj =
document.getElementById('pivotfieldlist1').ej2_instances[0];
      let pivotGridObj =
document.getElementById('pivotview_flist').ej2_instances[0];
      fieldListObj.updateView(pivotGridObj);
    }
  }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
    @import "@syncfusion/ej2-vue-pivotview/styles/material.css";
    #pivotfieldlist1 {
      width: 400px;
      margin-top: 20px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs72" %}

FieldListRefreshed

The event fieldListRefreshed is triggered whenever there is any change done in the field list UI. It has following parameter - dataSourceSettings and pivotValues. It allows user to identify each field list update. This event is applicable only for static field list.

For example, if we perform a sort operation within the field list, the field list will be refreshed. The fieldListRefreshed event will be triggered at that time and the user can perform custom operation inside that event.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"
:fieldListRefreshed="fieldListRefreshed"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
```

```
        pivotview: [FieldList]
    },
  methods: {
    fieldListRefreshed: function() {
      //Triggers, whenever field list get refreshed.
    }
  }
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs73" %}

OnFieldDropped

The event onFieldDropped fires whenever a field is dropped in an axis. It has following parameters - droppedAxis, droppedField and dataSourceSettings. In this illustration, we have modified the droppedField caption through this event at runtime.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"
:onFieldDropped="onFieldDropped"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
```

```
  provide: {
        pivotview: [FieldList]
    },
  methods: {
    onFieldDropped: function(args: FieldDroppedEventArgs) {
      //Triggers, whenever field is dropped in an axis.
        args.droppedField.caption = args.droppedField.name + " --> " +
args.droppedAxis;
    }
  }
}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs74" %}

ActionBegin

The event actionBegin triggers when the UI actions such as sorting, filtering, aggregation or edit calculated field, that are present in the field list UI begin. This allows user to identify the current action being performed at runtime. It has the following parameters:

- dataSourceSettings: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- actionName: It holds the name of the current action began. The following are the UI actions and their names:

| Action | Action Name|

|------|-------------|

| Sort icon| Sort field|

| Filter icon| Filter field|

| Aggregation (Value type drop down and menu)| Aggregate field|

| Edit icon| Edit calculated field|

| Calculated field button| Open calculated field dialog|

| Field list| Open field list|

| Field list tree – Sort icon| Sort field tree|

- fieldInfo: It holds the selected field information.

Note: This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- cancel: It allows user to restrict the current action.

Copyright © 2001 -2024 Syncfusion Inc.

In the below sample, opening pop-up field list can be restricted by setting the **args.cancel** option to **true** in the actionBegin event.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList" :actionBegin="actionBegin">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList, PivotActionBeginEventArgs } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
  methods: {
    actionBegin: function (args: PivotActionBeginEventArgs) {
      if (args.actionName == 'Open field list') {
        args.cancel = true;
      }
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs75" %}

Copyright © 2001 -2024 Syncfusion Inc.

ActionComplete

The event actionComplete triggers when the UI actions such as sorting, filtering, aggregation or edit calculated field, that are present in the field list UI, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- dataSourceSettings: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- actionName: It holds the name of the current action completed. The following are the UI actions and their names:

| Action | Action Name|

|------|------------|

| Sort icon| Field sorted|

| Filter icon| Field filtered|

| Aggregation(Value type drop down and menu)| Field aggregated|

| Edit icon| Calculated field edited|

| Calculated field button| Calculated field applied|

| Field list| Field list closed|

| Field list tree – Sort icon| Field tree sorted|

- fieldInfo: It holds the selected field information.

Note: This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- actionInfo: It holds the unique information about the current UI action. For example, if sorting is completed, the event argument contains information such as sort order and the field name.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"
:actionComplete="actionComplete"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList, PivotActionCompleteEventArgs } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
  methods: {
    actionComplete: function (args: PivotActionCompleteEventArgs) {
      if (args.actionName == 'Field list closed') {
        // Triggers when the field list dialog is closed.
      }
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs76" %}

ActionFailure

The event actionFailure triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- actionName: It holds the name of the current action failed. The following are the UI actions and their names:

| Action | Action Name|

|------|------------|

| Sort icon| Sort field|

| Filter icon| Filter field|

| Aggregation (Value type drop down and menu)| Aggregate field|

| Edit icon| Edit calculated field|

| Calculated field button| Open calculated field dialog|

Copyright © 2001 -2024 Syncfusion Inc.

| Field list | Open field list |

| Field list tree – Sort icon | Sort field tree |

- errorInfo: It holds the error information of the current UI action.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"
:actionFailure="actionFailure"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList, PivotActionFailureEventArgs } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
  methods: {
    actionFailure: function (args: PivotActionFailureEventArgs) {
      if (args.actionName == 'Open field list') {
        // Triggers when the current UI action fails to achieve the desired
result.
      }
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
```

```
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs77" %}

See Also

- Customize the icons for pivot table

Defer update in Vue Pivotview component

Defer layout update support allows to update the pivot table component only on demand. On enabling this feature, end user can drag-and-drop fields between row, column, value and filter axes, apply sorting and filtering inside the Field List, resulting in change of pivot report alone but not the pivot table values. Once all operations are performed and on clicking the "Apply" button in the Field List, pivot table will start to update with the last modified report. This also helps in bringing better performance in pivot table component rendering.

The field list can be displayed in two different formats to interact with pivot table. They are:

- **In-built Field List (Popup)**: To display the field list icon in pivot table UI to invoke the built-in dialog.
- **Stand-alone Field List (Fixed)**: To display the field list in a static position within a web page.

In-built Field List (Popup)

To enable deferred updates in the pivot table, set the property allowDeferLayoutUpdate in pivot table as **true**. To make a note, the defer update option can be controlled only via Field List during runtime.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"
:allowDeferLayoutUpdate="allowDeferLayoutUpdate"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
```

```
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true,
      allowDeferLayoutUpdate: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs25" %}

Stand-alone Field List (Fixed)

The field list can be rendered in a static position, anywhere in web page layout, like a separate component. To do so, you need to set renderMode property to **Fixed** in pivot field list.

To enable deferred updates in the static fieldlist, set the property allowDeferLayoutUpdate in pivot field list as **true**. To make a note, the defer update option can be controlled only via Field List during runtime.

To make field list interact with pivot table, you need to use the **UpdateView** and **Update** methods for data source update in both field list and pivot table simultaneously.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview_flist" :height="height"
:allowDeferLayoutUpdate="allowDeferLayoutUpdate"></ejs-pivotview>
        <ejs-pivotfieldlist id="pivotfieldlist1"
:allowDeferLayoutUpdate="allowDeferLayoutUpdate"
:dataSourceSettings="dataSourceSettings"
:enginePopulated="fieldEnginePopulated" :renderMode="renderMode"></ejs-
pivotfieldlist>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotFieldListPlugin, FieldList } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(PivotFieldListPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
```

```
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      renderMode: "Fixed",
      allowDeferLayoutUpdate: true
    }
  },
  methods: {
    fieldEnginePopulated: function(args) {
      let fieldlistObj =
document.getElementById('pivotfieldlist1').ej2_instances[0];
      let pivotGridObj =
document.getElementById('pivotview_flist').ej2_instances[0];
      if (fieldlistObj.isRequiredUpdate) {
            fieldlistObj.updateView(pivotGridObj);
        }
        pivotGridObj.notify('ui-update', pivotGridObj);
        fieldlistObj.notify('tree-view-update', fieldlistObj);
    }
  }
}
</script>
<style>
    @import "@syncfusion/ej2-vue-pivotview/styles/material.css";
    #pivotfieldlist1 {
      width: 400px;
      margin-top: 20px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs26" %}

Performance

<!-- markdownlint-disable MD036 -->

Virtual scrolling in Vue Pivotview component

Virtual Scrolling

The virtual scrolling option allows you to load the large amounts of data without performance degradation by rendering rows and columns only in the content viewport. The data will refresh dynamically on vertical or horizontal scroll. This feature can be enabled by setting the enableVirtualization property to **true**.

To use the virtual scrolling feature, inject the VirtualScroll module in to the pivot table.

APP.VUE

```
<template>
    <div id="app">
```

```
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :enableVirtualization="enableVirtualization"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, VirtualScroll, IDataSet } from "@syncfusion/ej2-
vue-pivotview";
Vue.use(PivotViewPlugin);
/* tslint:disable */
let date1: number;
let date2: number;
let isInit: boolean;
function data(count: number) {
  let result: Object[] = [];
  let dt: number = 0;
  for (let i: number = 1; i < count + 1; i++) {
    dt++;
    let round: string;
    let toString: string = i.toString();
    if (toString.length === 1) {
      round = "0000" + i;
    } else if (toString.length === 2) {
      round = "000" + i;
    } else if (toString.length === 3) {
      round = "00" + i;
    } else if (toString.length === 4) {
      round = "0" + i;
    } else {
      round = toString;
    }
    result.push({
      ProductID: "PRO-" + round,
      Year: "FY " + (dt + 2013),
      Price: Math.round(Math.random() * 5000) + 5000,
      Sold: Math.round(Math.random() * 80) + 10
    });
    if (dt / 4 == 1) {
        dt = 0;
    }
  }
  return result;
}
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: data(1000) as IDataSet[],
        enableSorting: false,
        expandAll: true,
        formatSettings: [{ name: 'Price', format: 'C0' }],
        rows: [{ name: 'ProductID' }],
        columns: [{ name: 'Year' }],
        values: [{ name: 'Price', caption: 'Unit Price' }, { name: 'Sold',
caption: 'Unit Sold' }]
      },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      height: 350,
      enableVirtualization: true
    }
  },
  provide: {
        pivotview: [VirtualScroll]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs33" %}

Limitations for virtual scrolling

- In virtual scrolling, the columnWidth property in gridSettings should be in pixel and percentage values are not accepted.
- Resizing columns, setting width to individual columns which affects the calculation used to pick the correct page on scrolling.
- Grouping, which takes additional time to splitting the raw items into the provided format.
- Date Formatting, which takes additional time to convert date format.
- Date Formatting with sorting, here additionally full date time format should be framed to perform sorting along with the provided date format which lags the performance.
- When using OLAP data, subtotals and grandtotals are only displayed when measures are bound at the last position in the rows or columns axis. Otherwise, the data from the pivot table will be shown without summary totals.
- Even if virtual scrolling is enabled, not only is the current view port data retrieved, but also the data for the immediate previous page and the immediate next page. As a result, when the end user scrolls slightly ahead or behind, the next or previous page data is displayed immediately without requiring a refresh. **Note:** If the pivot table's width and height are large, the loading data count in the current, previous, and next view ports (pages) will also increase, affecting performance.

Data Compression

This property is applicable only for the relational data source.

When we bind one million raw data, the pivot table will process all raw data to generate aggregated data during initial rendering and report manipulation. But in data compression, the data will be compressed based on the uniqueness of the raw data, and unique records will be provided as input for the Pivot Table. The compressed data will be used for further operations at all times, reducing the looping complexity and improving the performance of the pivot table. For example, if the pivot table is connected to one million raw data aggregated to 1,000 unique data means, it will be rendered within 3 seconds rather than 10 seconds. You can enable this option by using the allowDataCompression property along with enableVirtualization property.

This options will only function when the virtual scrolling is enabled.

APP.VUE

```
<template>
```

```
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :allowDataCompression="allowDataCompression"
:enableVirtualization="enableVirtualization"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, VirtualScroll, IDataSet } from "@syncfusion/ej2-
vue-pivotview";
Vue.use(PivotViewPlugin);
/* tslint:disable */
let date1: number;
let date2: number;
let isInit: boolean;
function data(count: number) {
  let result: Object[] = [];
  let dt: number = 0;
  for (let i: number = 1; i < count + 1; i++) {
    dt++;
    let round: string;
    let toString: string = i.toString();
    if (toString.length === 1) {
      round = "0000" + i;
    } else if (toString.length === 2) {
      round = "000" + i;
    } else if (toString.length === 3) {
      round = "00" + i;
    } else if (toString.length === 4) {
      round = "0" + i;
    } else {
      round = toString;
    }
    result.push({
      ProductID: 'PRO-' + (i % 1000),
      Year: "FY " + (dt + 2013),
      Price: Math.round(Math.random() * 5000) + 5000,
      Sold: Math.round(Math.random() * 80) + 10
    });
    if (dt / 4 == 1) {
        dt = 0;
    }
  }
  return result;
}
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: data(1000) as IDataSet[],
        enableSorting: false,
        expandAll: true,
        formatSettings: [{ name: 'Price', format: 'C0' }],
        rows: [{ name: 'ProductID' }],
        columns: [{ name: 'Year' }],
        values: [{ name: 'Price', caption: 'Unit Price' }, { name: 'Sold',
caption: 'Unit Sold' }]
```

```
      },
      height: 350,
      allowDataCompression: true,
      enableVirtualization: true
    }
  },
  provide: {
        pivotview: [VirtualScroll]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs34" %}

Limitations during data compression

- The following aggregation types will not be supported.
- Average
- Populationstdev
- Samplestdev
- Populationvar
- Samplevar
- If you use any of the aggregations above, it will result in an aggregation type **“Sum”**.
- Distinctcount will act as **“Count”** aggregation type.
- In the calculated field, an existing field can be inserted without altering its default aggregation type Even if we change it, it would use the default aggregation type back for calculation.

Virtual scrolling for static field list

Virtual scrolling automatically works with "Popup" field list on setting the enableVirtualization property in the Pivot Table to **true**. Incase of static field list, which act as a separate component, user need to enable enableVirtualization property in the Pivot Table and also pass the report information to pivot table instance via the load event of the field list.

APP.VUE

```
<template>
  <div id="app">
      <ejs-pivotview id="pivotview_flist" :height="height"
:gridSettings="gridSettings" :enginePopulated="enginePopulated"
:width="width" :enableVirtualization="enableVirtualization"
      ></ejs-pivotview>
      <ejs-pivotfieldlist id="pivotfieldlist1"
:dataSourceSettings="dataSourceSettings"
:enginePopulated="fieldEnginePopulated" :load="load" :dataBound="dataBound"
:allowCalculatedField="allowCalculatedField" :renderMode="renderMode"
      ></ejs-pivotfieldlist>
  </div>
</template>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { PivotViewPlugin, PivotFieldListPlugin, FieldList, CalculatedField,
VirtualScroll} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(PivotFieldListPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      renderMode: "Fixed",
      gridSettings: { columnWidth: 140 },
      allowCalculatedField: true,
      enableVirtualization: true,
      width: "99%",
    }
  },
  methods: {
    fieldEnginePopulated: function() {
      let fieldListObj =
document.getElementById('pivotfieldlist1').ej2_instances[0];
      let pivotGridObj =
document.getElementById('pivotview_flist').ej2_instances[0];
      fieldListObj.updateView(pivotGridObj);
    },
    dataBound: function () {
      let pivotObj =
document.getElementById('pivotview_flist').ej2_instances[0];
      pivotObj.tooltip.destroy();
      pivotObj.refresh();
    },
    load: function () {
      let pivotObj =
document.getElementById('pivotview_flist').ej2_instances[0];
      let fieldListObj =
document.getElementById('pivotfieldlist1').ej2_instances[0];
      pivotObj.dataSourceSettings = fieldListObj.dataSourceSettings;
      pivotObj.updatePageSettings(true);
      fieldListObj.pageSettings = pivotObj.pageSettings;
    },
    enginePopulated: function () {
        let pivotObj =
document.getElementById('pivotview_flist').ej2_instances[0];
        let fieldListObj =
document.getElementById('pivotfieldlist1').ej2_instances[0];
        if (fieldListObj && pivotObj) {
          fieldListObj.update(pivotObj);
```

```
        }
    }
  },
  provide: {
    pivotview: [CalculatedField, FieldList, VirtualScroll],
  },
}
</script>
<style>
    @import "@syncfusion/ej2-vue-pivotview/styles/material.css";
    #pivotfieldlist1 {
      width: 400px;
      margin-top: 20px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs260" %}

Paging in Vue Pivotview component

Paging allows you to load large amounts of data that can be divided and displayed page by page in the pivot table. It can be enabled by setting the enablePaging property to **true**. It can be configured at code-behind by using the pageSettings property, during initial rendering of the component. The properties required are:

- currentRowPage: Allows user to set the current row page number to be displayed in the pivot table.
- currentColumnPage: Allows user to set the current column page number to be displayed in the pivot table.
- rowPageSize: Allows user to set the total number of records to be displayed on each page of the pivot table's row axis.
- columnPageSize: Allows user to set the total number of records to be displayed on each page of the pivot table's column axis.

Pager UI

When paging is enabled, a built-in pager UI appears at the bottom of the pivot table, allowing you to change the current page in the row and column axes by navigating to a desired page using the navigation buttons or an input text box, as well as change the page size via dropdown at runtime.

You can also change the position, visibility, compact view, and template of the row and column pagers by using the pagerSettings.

In order to see and use the pager UI, insert the Pager module into the pivot table using the provide section.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:enablePaging="enablePaging" :pageSettings="pageSettings"
:pagerSettings="pagerSettings" :gridSettings="gridSettings"> </ejs-
pivotview>
```

```
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, Pager } from "@syncfusion/ej2-vue-pivotview";
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
Vue.use(PivotViewPlugin);
let remoteData: DataManager = new DataManager({
  url: "https://bi.syncfusion.com/northwindservice/api/orders",
  adaptor: new WebApiAdaptor(),
  crossDomain: true
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: remoteData as DataManager,
        expandAll: true,
        filters: [],
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
        formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
        values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
      },
      height: 350,
      enablePaging: true,
      pageSettings: {
        rowPageSize: 10,
        columnPageSize: 5,
        currentColumnPage: 1,
        currentRowPage: 1
      },
      pagerSettings: {
        position: 'Bottom',
        enableCompactView: false,
        showColumnPager: true,
        showRowPager: true,
        columnPageSizes: [5, 10, 20, 50, 100],
        rowPageSizes: [10, 50, 100, 200],
        isInversed: false,
        showColumnPageSize: true,
        showRowPageSize: true
      },
      gridSettings: { columnWidth: 120 },
    }
  },
  provide: {
    pivotview: [Pager]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs25" %}

Show pager UI at top or bottom

You can display the pager UI at top or bottom of the pivot table by using the position property. To show the pager UI at top of the pivot table, set the position property in pagerSettings to **Top**.

By default, the pager UI appears at the bottom of the pivot table.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:enablePaging="enablePaging" :pageSettings="pageSettings"
:pagerSettings="pagerSettings" :gridSettings="gridSettings"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, Pager } from "@syncfusion/ej2-vue-pivotview";
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
Vue.use(PivotViewPlugin);
let remoteData: DataManager = new DataManager({
  url: "https://bi.syncfusion.com/northwindservice/api/orders",
  adaptor: new WebApiAdaptor(),
  crossDomain: true
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: remoteData as DataManager,
        expandAll: true,
        filters: [],
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
        formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
        values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
      },
      height: 350,
      enablePaging: true,
      pageSettings: {
        rowPageSize: 10,
        columnPageSize: 5,
        currentColumnPage: 1,
        currentRowPage: 1
      },
      pagerSettings: {
        position: 'Top'
      },
      gridSettings: { columnWidth: 120 },
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
  provide: {
    pivotview: [Pager]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs26" %}

Inverse pager

Toggles and displays row and column pager. To show the column pager on the left side of the pager UI, set the isInversed property in pagerSettings to **true**.

By default, the row pager is displayed on the left side of the pager UI, while the column pager is displayed on the right side.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:enablePaging="enablePaging" :pageSettings="pageSettings"
:pagerSettings="pagerSettings" :gridSettings="gridSettings"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, Pager } from "@syncfusion/ej2-vue-pivotview";
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
Vue.use(PivotViewPlugin);
let remoteData: DataManager = new DataManager({
  url: "https://bi.syncfusion.com/northwindservice/api/orders",
  adaptor: new WebApiAdaptor(),
  crossDomain: true
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: remoteData as DataManager,
        expandAll: true,
        filters: [],
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
        formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
        values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
      },
      height: 350,
      enablePaging: true,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      pageSettings: {
        rowPageSize: 10,
        columnPageSize: 5,
        currentColumnPage: 1,
        currentRowPage: 1
      },
      pagerSettings: {
        isInversed: true
      },
      gridSettings: { columnWidth: 120 },
    }
  },
  provide: {
    pivotview: [Pager]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs27" %}

Compact view

By hiding all except the previous and next navigation buttons, the pager UI can be displayed with the absolute minimum of paging options. The compact view can be enabled by setting the enableCompactView property in pagerSettings to **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:enablePaging="enablePaging" :pageSettings="pageSettings"
:pagerSettings="pagerSettings" :gridSettings="gridSettings"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, Pager } from "@syncfusion/ej2-vue-pivotview";
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
Vue.use(PivotViewPlugin);
let remoteData: DataManager = new DataManager({
  url: "https://bi.syncfusion.com/northwindservice/api/orders",
  adaptor: new WebApiAdaptor(),
  crossDomain: true
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: remoteData as DataManager,
        expandAll: true,
        filters: [],
```

```
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
        formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
        values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
      },
      height: 350,
      enablePaging: true,
      pageSettings: {
        rowPageSize: 10,
        columnPageSize: 5,
        currentColumnPage: 1,
        currentRowPage: 1
      },
      pagerSettings: {
        enableCompactView: true
      },
      gridSettings: { columnWidth: 120 },
    }
  },
  provide: {
    pivotview: [Pager]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs28" %}

Show or hide paging option

By using the showRowPager and showColumnPager properties in pagerSettings, you can show or hide row and column pager separately in the pager UI.

In the following example, row pager has been disabled by setting the showRowPager property in pagerSettings to **false**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:enablePaging="enablePaging" :pageSettings="pageSettings"
:pagerSettings="pagerSettings" :gridSettings="gridSettings"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, Pager } from "@syncfusion/ej2-vue-pivotview";
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
Vue.use(PivotViewPlugin);
let remoteData: DataManager = new DataManager({
```

```
  url: "https://bi.syncfusion.com/northwindservice/api/orders",
  adaptor: new WebApiAdaptor(),
  crossDomain: true
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: remoteData as DataManager,
        expandAll: true,
        filters: [],
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
        formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
        values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
      },
      height: 350,
      enablePaging: true,
      pageSettings: {
        rowPageSize: 10,
        columnPageSize: 5,
        currentColumnPage: 1,
        currentRowPage: 1
      },
      pagerSettings: {
        showRowPager: false
      },
      gridSettings: { columnWidth: 120 },
    }
  },
  provide: {
    pivotview: [Pager]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs29" %}

Show or hide page size

By using the showRowPageSize and showColumnPageSize properties in pagerSettings, you can show or hide "Rows per page" and "Columns per page" dropdown menu. The dropdown menu contains a list of pre-defined or user-defined page sizes, which will be displayed in the "Rows per page" and "Columns per page" dropdowns, allowing you to change the page size for the row and column axes at runtime.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:enablePaging="enablePaging" :pageSettings="pageSettings"
```

```
:pagerSettings="pagerSettings" :gridSettings="gridSettings"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, Pager } from "@syncfusion/ej2-vue-pivotview";
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
Vue.use(PivotViewPlugin);
let remoteData: DataManager = new DataManager({
  url: "https://bi.syncfusion.com/northwindservice/api/orders",
  adaptor: new WebApiAdaptor(),
  crossDomain: true
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: remoteData as DataManager,
        expandAll: true,
        filters: [],
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
        formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
        values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
      },
      height: 350,
      enablePaging: true,
      pageSettings: {
        rowPageSize: 10,
        columnPageSize: 5,
        currentColumnPage: 1,
        currentRowPage: 1
      },
      pagerSettings: {
        showColumnPageSize: false,
        showRowPageSize: false
      },
      gridSettings: { columnWidth: 120 },
    }
  },
  provide: {
    pivotview: [Pager]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs30" %}

Copyright © 2001 -2024 Syncfusion Inc.

Customize page size

By using the rowPageSizes and columnPageSizes properties in pagerSettings, you can specify a set of desired page sizes, which will be displayed in the "Rows per page" and "Columns per page" dropdowns, allowing you to change the page size for the row and column axes at runtime.

By default, the "Rows per page" dropdown have pre-defined page sizes of **10, 50, 100, and 200**, while the "Columns per page" dropdown have pre-defined page sizes of **5, 10, 20, 50, and 100**.

In the following example, the "Rows per page" dropdown is set with user-defined page sizes of **10, 20, 30, 40, and 50** and the "Columns per page" dropdown is set with user-defined page sizes of **5, 10, 15, 20, and 30**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:enablePaging="enablePaging" :pageSettings="pageSettings"
:pagerSettings="pagerSettings" :gridSettings="gridSettings"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, Pager } from "@syncfusion/ej2-vue-pivotview";
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
Vue.use(PivotViewPlugin);
let remoteData: DataManager = new DataManager({
  url: "https://bi.syncfusion.com/northwindservice/api/orders",
  adaptor: new WebApiAdaptor(),
  crossDomain: true
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: remoteData as DataManager,
        expandAll: true,
        filters: [],
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
        formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
        values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
      },
      height: 350,
      enablePaging: true,
      pageSettings: {
        rowPageSize: 10,
        columnPageSize: 5,
        currentColumnPage: 1,
        currentRowPage: 1
      },
      pagerSettings: {
        columnPageSizes: [5, 10, 15, 20, 30],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        rowPageSizes: [10, 20, 30, 40, 50]
      },
      gridSettings: { columnWidth: 120 },
    }
  },
  provide: {
    pivotview: [Pager]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs31" %}

Template

The template property allows to change the appearance of the pager UI by displaying user-defined HTML elements instead of built-in HTML elements.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview-template" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:enablePaging="enablePaging" :pageSettings="pageSettings"
:pagerSettings="pagerSettings" :gridSettings="gridSettings"
:dataBound="onDataBound"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, Pager } from "@syncfusion/ej2-vue-pivotview";
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
import { Pager as GridPager } from '@syncfusion/ej2-grids';
import { isNullOrUndefined } from "@syncfusion/ej2-base";
Vue.use(PivotViewPlugin);
let remoteData: DataManager = new DataManager({
  url: "https://bi.syncfusion.com/northwindservice/api/orders",
  adaptor: new WebApiAdaptor(),
  crossDomain: true
});
let rowPager: GridPager;
let columnPager: GridPager;
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: remoteData as DataManager,
        expandAll: true,
        filters: [],
        columns: [{ name: 'ProductName', caption: 'Product Name' }],
        rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
        formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
```

```
        values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
      },
      height: 350,
      enablePaging: true,
      pageSettings: {
        rowPageSize: 10,
        columnPageSize: 5,
        currentColumnPage: 1,
        currentRowPage: 1
      },
      pagerSettings: {
        template: '#template'
      },
      gridSettings: { columnWidth: 120 },
    }
  },
  methods: {
    onDataBound: function() {
      this.updateTemplate()
    },
    updateTemplate() {
        let pivotObj = document.getElementById('pivotview-
template').ej2_instances[0];;
        if (!isNullOrUndefined(rowPager)) {
            rowPager.destroy();
            rowPager = null;
        }
        rowPager = new GridPager({
            pageSize: pivotObj.pageSettings.rowPageSize,
            totalRecordsCount: pivotObj.engineModule.rowCount,
            currentPage: pivotObj.pageSettings.currentRowPage,
            pageCount: 5,
            click: this.rowPageClick
        });
        rowPager.appendTo('#row-pager');
        if (!isNullOrUndefined(columnPager)) {
            columnPager.destroy();
            columnPager = null;
        }
        columnPager = new GridPager({
            pageSize: pivotObj.pageSettings.columnPageSize,
            totalRecordsCount: pivotObj.engineModule.columnCount,
            currentPage: pivotObj.pageSettings.currentColumnPage,
            pageCount: 5,
            click: this.columnPageClick
        });
        columnPager.appendTo('#column-pager');
    },
    rowPageClick(args: any) {
        let pivotObj = document.getElementById('pivotview-
template').ej2_instances[0];;
        pivotObj.pageSettings.currentRowPage = args.currentPage;
        pivotObj.refreshData();
    },
    columnPageClick(args: any) {
```

```
        let pivotObj = document.getElementById('pivotview-
template').ej2_instances[0];;
        pivotObj.pageSettings.currentColumnPage = args.currentPage;
        pivotObj.refreshData();
    }
  },
  provide: {
    pivotview: [Pager]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs32" %}

State persistence in Vue Pivotview component

State persistence allows user to maintain the current state of the component along with its report bounded in the browser local storage (cookie). Even if the browser is refreshed or if you move to the next page within the browser, components state will be persisted. State persistence stores the Pivot Table object in the local storage when enablePersistence property in pivot table is set to **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showGroupingBar="showGroupingBar"
:enablePersistence="enablePersistence"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar } from "@syncfusion/ej2-vue-
pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showGroupingBar: true,
      enablePersistence: true
    }
```

```
  },
  provide: {
        pivotview: [GroupingBar]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs234" %}

Save and Load Pivot Layout

You can save the current layout of the pivot table by using getPersistData in string format. The saved layout can be loaded to pivot table any time by passing the saved data as a parameter to loadPersistData method in the pivot table.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="save-btn" :isPrimary="isPrimary" v-
on:click.native="saveBtnClick">Save Layout</ejs-button>
        <ejs-button id="load-btn" :isPrimary="isPrimary" v-
on:click.native="loadBtnClick">Load Layout</ejs-button>
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:showGroupingBar="showGroupingBar" :enablePersistence="enablePersistence">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotFieldListPlugin, FieldList, GroupingBar }
from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(PivotFieldListPlugin);
Vue.use(ButtonPlugin);
let layout: string;
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
```

```
      showGroupingBar: true,
      enablePersistence: true,
      isPrimary: true
    }
  },
  provide: {
       pivotview: [GroupingBar]
   }
  methods: {
    saveBtnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      this.layout = pivotGridObj.getPersistData();
    },
    loadBtnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.loadPersistData(this.layout);
    }
  }
}
</script>
<style>
    @import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs235" %}

<!-- markdownlint-disable MD012 -->

Row and column in Vue Pivotview component

Width and Height

Allows end user to set the pivot table's height and width by using height and width properties in pivot table respectively. The supported formats to set height and width properties are,

- Pixel: For example - 100, 200, "100px", "200px".
- Percentage: For example - "100%", "200%".
- Auto: It is applicable for height property alone in-order to render the pivot table beyond its parent container height without vertical scrollbar. The parent container here would show its vertical scrollbar as soon as the component reaches beyond its dimension.

The pivot table will not be displayed less than **400px**, since it's the minimum width of the component.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :width="width"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
```

```
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: '340px',
      width: 550
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs205" %}

Row Height

Allows end user to set the height of each pivot table rows commonly using the rowHeight property in gridSettings.

By default, the rowHeight property is set as **36** pixels for desktop layout and **48** pixels for mobile layout.

The height of the column headers alone may vary when grouping bar feature is enabled.

In the below code sample, the rowHeight property is set as **60** pixels.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
```

```
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
          rowHeight: 60
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs206" %}

Column Width

Allows end user to set the width of each pivot table columns commonly using the columnWidth property in gridSettings.

By default, the columnWidth property is set as **110** pixels to each columns except the first column. For first column, **250** pixels and **200** pixels are set respectively with and without grouping bar.

In the below example, the columnWidth property is set as **200** pixels.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
```

```
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
          columnWidth: 120
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs207" %}

Reorder

Allows end user to reorder a particular column header from one index to another index within the pivot table through drag-and-drop option. It can be enabled by setting the allowReordering property in gridSettings to **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
          allowReordering: true
      }
    }
  }
```

```
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs208" %}

Column Resizing

Allows end user to resize the columns by clicking and dragging the right edge of the column header. While dragging, the width of the respective column will be resized immediately. To enable column resizing option, set the allowResizing property in gridSettings to **true**.

By default, the column resizing option is enabled.

In RTL mode, user can click and drag the left edge of the header cell to resize the column.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
          allowResizing: true
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs209" %}

Text Wrap

Allows end user to wrap the cell content to the next line when it exceeds the boundary of the cell width. To enable text wrap, set the allowTextWrap property in gridSettings to **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
          allowTextWrap: true
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs210" %}

Text Align

Allows end user to align the content of the pivot table's row and column headers and value cells by using the textAlign and headerTextAlign properties in the columnRender event under gridSettings. The following alignments are:

Copyright © 2001 -2024 Syncfusion Inc.

- Left - It allows the content to be positioned on the left.
- Right - It allows the content to be positioned on the right.
- Center - It allows the content to be positioned in the middle.
- Justify - It allows the content to be as flexible as possible, when the cell does not occupy the entire available area.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
        columnRender: function(args){
          if(args.stackedColumns[0]){
            // Content for the row headers is right-aligned here.
            args.stackedColumns[0].textAlign="Right";
          }
          if(args.stackedColumns[1]){
            // Content for the column header "FY 2015" is center-aligned
here.
            args.stackedColumns[1].textAlign = 'Center';
          }
          if(args.stackedColumns[1] && (args.stackedColumns[1].columns[0] as
any)){
            // Content for the column header "Q1" is right-aligned here.
            (args.stackedColumns[1].columns[0] as any).textAlign = 'Right';
          }
          if(args.stackedColumns[1] && args.stackedColumns[1].columns[0] &&
(args.stackedColumns[1].columns[0] as any).columns[0]){
            // Content for the value header "Units Sold" is right-aligned
here.
```

```
            (args.stackedColumns[1].columns[0] as
any).columns[0].headerTextAlign = 'Right';
          }
          if(args.stackedColumns[1] && args.stackedColumns[1].columns[0] &&
(args.stackedColumns[1].columns[0] as any).columns[0]){
            // Content for the values are left-aligned here.
            (args.stackedColumns[1].columns[0] as any).columns[0].textAlign
= 'Left';
          }
        }
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs211" %}

AutoFit

Allows the user to fit the Pivot Table columns as wide as the content of the cell without wrapping. It auto fits all of the Pivot Table columns by invoking the autoFitColumns method from the grid instance.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:dataBound="ondataBound"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
    }
  },
```

```
  methods: {
    ondataBound: function() {
     let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
     pivotGridObj.grid.autoFitColumns();
    }
  },
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs212" %}

The minimum width of 250 pixels is set by default with the grouping bar UI for the first column and cannot be reduced further. So, when the grouping bar is enabled, one can auto fit the Pivot Table columns by calling the autoFitColumns method from the grid instance with the parameter contained pivot table columns field name excluding first column.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:showGroupingBar="showGroupingBar"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar } from "@syncfusion/ej2-vue-
pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showGroupingBar: true
    }
  },
  methods: {
    trend: function() {
```

```
     let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
     if (pivotGridObj.showGroupingBar) {
        let columns = [];
        for (let i = 1; i < (pivotGridObj.grid).columnModel.length; i++) {
          columns.push((pivotGridObj.grid).columnModel[i].field);
        }
        pivotGridObj.grid.autoFitColumns(columns);
      }
    }
  },
  provide: {pivotview: [GroupingBar]}
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs213" %}

Autofit specific columns

During initial rendering, the parameter autoFit in the columnRender event under gridSettings can be set to **true** to auto fit specific columns.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
        columnRender: function(args){
          for (let i = 0; i < args.columns.length; i++) {
                args.columns[i].autoFit = true;
```

```
            }
        }
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs214" %}

Grid Lines

Allows end user to display cell border for each cells using gridLines property in gridSettings.

Available mode of grid lines are:

| Modes | Actions |

|-------|---------|

| Both | Displays both the horizontal and vertical grid lines.|

| None | No grid lines are displayed.|

| Horizontal | Displays the horizontal grid lines only.|

| Vertical | Displays the vertical grid lines only.|

| Default | Displays grid lines based on the theme.|

By default, pivot table renders grid lines in **Both** mode.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
```

```
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
          gridLines: 'Vertical'
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs215" %}

Selection

Selection provides an option to highlight a row or a column or a cell. It can be done through simple mouse down or arrow keys. To enable selection in the pivot table, set the allowSelection property in gridSettings to **true**.

The pivot table supports two types of selection that can be set using type property in selectionSettings. The selection types are:

- **Single**: It is set by default, and it only allows selection of a single row or a column or a cell.
- **Multiple**: Allows you to select multiple rows or columns or cells.

To perform multi-selection, press and hold "CTRL" key and click the desired rows or cells. To select range of rows or cells, press and hold the "SHIFT" key and click the rows or columns or cells.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
          allowSelection: true,
          selectionSettings: { type: 'Multiple'}
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs216" %}

Selection Mode

The pivot table supports four types of selection mode that can be set using mode in selectionSettings. The selection modes are:

- **Row**: It is set by default, and allows user to select only rows.
- **Column**: Allows user to select only columns.
- **Cell**: Allows user to select only cells.
- **Both**: Allows user to select rows and columns at the same time.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      },
      height: 350,
      gridSettings: {
          allowSelection: true,
          selectionSettings: { mode: 'Both'}
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs217" %}

Cell Selection Mode

The pivot table supports two types of cell selection mode that can be set using cellSelectionMode in selectionSettings. The cell selection modes are:

- **Flow**: It is set by default. The range of cells are selected between the start index and end index that includes in-between cells of rows.
- **Box**: Range of cells are selected from the start and end column indexes that includes in-between cells of rows within the range.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
```

```
            allowSelection: true,
            selectionSettings: { cellSelectionMode: 'Box', type: 'Multiple',
mode: 'Cell'}
        }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs218" %}

Cell selection requires mode property in selectionSettings to be **Cell** or **Both**, and type property should be **Multiple**.

Changing background color of the selected cell

The background-color of the selected cell can be changed using built-in CSS names. To do so, please refer to the code sample below, which shows that the selected cells are changed to a **green yellow** color.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
          allowSelection: true,
          selectionSettings: { cellSelectionMode: 'Box', type: 'Multiple',
mode: 'Cell'}
      }
    }
```

```
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
/* csslint ignore:start */
.e-pivotview .e-cellselectionbackground,
.e-pivotview .e-selectionbackground,
.e-pivotview .e-grid .e-rowsheader.e-selectionbackground,
.e-pivotview .e-grid .e-columnsheader.e-selectionbackground {
    background-color: greenYellow !important;
}
/* csslint ignore:end */
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs219" %}

Event

CellSelected

The event cellSelected is triggered when cell selection gets completed. It provides selected cells information with its corresponding column and row headers. It has following parameters - selectedCellsInfo, currentCell and target. This event allows user to view selected cells information and user can pass those selected cells information to any external component for data binding.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:cellSelected="cellSelected"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotCellSelectedEventArgs } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      },
      height: 350,
      gridSettings: {
        allowSelection: true,
        selectionSettings: { mode: 'Both', type: 'Multiple' }
      }
    }
  },
  methods: {
    cellSelected: function(args: PivotCellSelectedEventArgs) {
        //args.selectedCellsInfo -> get selected cells information.
        //args.pivotValues -> get the pivot values of the pivot table.
    },
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs220" %}

CellSelecting

The event cellSelecting triggers before cell gets selected gets completed. It provides selected cells information with its corresponding column and row headers. It has following parameters - currentCell, data and cancel.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:cellSelecting="cellSelecting"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PivotCellSelectedEventArgs } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
```

```
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
        allowSelection: true,
        selectionSettings: { mode: 'Both', type: 'Multiple' }
      }
    }
  },
  methods: {
    cellSelecting: function(args: PivotCellSelectedEventArgs) {
        //args.selectedCellsInfo -> get selected cells information.
        //args.pivotValues -> get the pivot values of the pivot table.
    },
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs221" %}

Clip Mode

The clip mode provides options to display its overflow cell content in the pivot table. It can be configured using the clipMode property in gridSettings. The pivot table supports three types of clip modes which are:

- **Clip**: Truncates the cell content when it overflows its area.
- **Ellipsis**: Displays ellipsis when the cell content overflows its area.
- **EllipsisWithTooltip**: Displays ellipsis when the cell content overflows its area, also it will display the tooltip while hover on ellipsis is applied.

By default, clipMode value is set to **Ellipsis**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
```

```
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
          clipMode: 'Clip'
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs222" %}

Cell Template

User can customize the pivot table cell element by using the cellTemplate property in pivot table The cellTemplate property accepts either an HTML string or the element's ID, which can be used to append additional HTML elements to showcase each cell with custom format.

In this demo, the revenue cost for each year is represented with trend icons.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height" :dataBound="trend"
:cellTemplate="cellTemplate"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { renewableEnergy } from './datasource.js';
Vue.use(PivotViewPlugin);
var cellTemplateVue = Vue.component("cellTemplate", {
    template: `<span class="template-wrap" v-
html=getCellContent(data)></span>`,
    data() {
        return {
            data: {}
        };
    },
    methods: {
            getCellContent: function (e) {
                return '<span class="tempwrap sb-icon-neutral pv-
icons"></span>';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            }
        }
    });
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: renewableEnergy,
        expandAll: true,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015', 'FY 2017', 'FY
2018'] }],
        formatSettings: [{ name: 'ProCost', format: 'C0' }],
        rows: [
            { name: 'Year', caption: 'Production Year' }
        ],
        columns: [
            { name: 'EnerType', caption: 'Energy Type' },
            { name: 'EneSource', caption: 'Energy Source' }
        ],
        values: [
            { name: 'ProCost', caption: 'Revenue Growth' }
        ],
        filters: []
      },
      height: 350,
      cellTemplate: function() {
        return { template: cellTemplateVue };
      },
    }
  },
  methods: {
    trend: function() {
      let pivotGridObj = (this.$refs.pivotview).ej2Instances;
      let rowIndx;
      var cTable = document.getElementsByClassName("e-table");
        var colLen = pivotGridObj.pivotValues[3].length;
        var cLen = cTable[3].children[0].children.length;
        var rLen = cTable[3].children[1].children.length;
        for (let k = 0; k < rLen; k++) {
            if (pivotGridObj.pivotValues[k] &&
pivotGridObj.pivotValues[k][0] !== undefined) {
                rowIndx = (pivotGridObj.pivotValues[k][0]).rowIndex;
                break;
            }
        }
        var rowHeaders =
[].slice.call(cTable[2].children[1].querySelectorAll('td'));
        var rows = pivotGridObj.dataSourceSettings.rows;
        if (rowHeaders.length > 1) {
            for (var i = 0, Cnt = rows; i < Cnt.length; i++) {
                var fields = {};
                var fieldHeaders = [];
                    for (var j = 0, Lnt = rowHeaders; j < Lnt.length; j++) {
                        var header = rowHeaders[j];
```

```
                        if (header.className.indexOf('e-gtot') === -1 &&
header.className.indexOf('e-rowsheader') > -1 &&
header.getAttribute('fieldname') === rows[i].name) {
                            var headerName =
rowHeaders[j].getAttribute('fieldname') + '_' + rowHeaders[j].textContent;
                            fields[rowHeaders[j].textContent] = j;
                            fieldHeaders.push(rowHeaders[j].textContent);
                        }
                    }
                    if (i === 0) {
                        for (var rnt = 0, Lnt = fieldHeaders; rnt <
Lnt.length; rnt++) {
                            if (rnt !== 0) {
                                var row = fields[fieldHeaders[rnt]];
                                var prevRow = fields[fieldHeaders[rnt - 1]];
                                for (var j = 0, ci = 1; j < cLen && ci <
colLen; j++, ci++) {
                                    var node =
cTable[3].children[1].children[row].childNodes[j];
                                    var prevNode =
cTable[3].children[1].children[prevRow].childNodes[j];
                                    var ri = node.getAttribute('index');
                                    var prevRi =
prevNode.getAttribute('index');
                                    if (ri <
pivotGridObj.pivotValues.length) {
                                        if
((pivotGridObj.pivotValues[prevRi][ci]).value >
(pivotGridObj.pivotValues[ri][ci]).value && node.querySelector('.tempwrap'))
{
                                            var trendElement =
node.querySelector('.tempwrap');
                                            trendElement.className =
trendElement.className.replace('sb-icon-neutral', 'sb-icon-loss');
                                        } else if
((pivotGridObj.pivotValues[prevRi][ci]).value <
(pivotGridObj.pivotValues[ri][ci]).value && node.querySelector('.tempwrap'))
{
                                            var trendElement =
node.querySelector('.tempwrap');
                                            trendElement.className =
trendElement.className.replace('sb-icon-neutral', 'sb-icon-profit');
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        },
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
.e-pivotview .e-columnsheader .tempwrap {
```

```
  display: none;
}
.e-pivotview .e-rowsheader .tempwrap {
  display: none;
}
@font-face {
  font-family: 'e-pivot';
  src:
      url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMjhUSmAAAAEoAAAAVmNtYXCs3q0zAAABkAAAAEpnbHlmdaI
tOwAAAegAAAE0aGVhZBRYEz0AAADQAAAANmhoZWEHmQNtAAAArAAAACRobXR4D7gAAAAAYAAAAA
QbG9jYQDAAHIAAAHcAAAACm1heHABDwBBAAABCAAAACBuYW1lc0cOBgAAAxwAAAIlcG9zdK9TctU
AAAVEAAAARwABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAAAAAABAABAAAAAQAAT8kba18
PPPUACwPoAAAAANin5zgAAAAA2KfnOAAAAAAD6gPoAAAACAACAAAAAAAAAAAEAAAAEADUAAQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPuAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4jToTQNS/2oAWgPoAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAIAAAADAAAAFAADAAEAAAAUAAQANgAAAAgACAACAADiNOI56E3//wAA4jTiOeh
N//8AAAAAAAAAAQAIAAgACAAAAAEAAwACAAAAAAAAACYAcgCaAAAAAQAAAAADTAPoABUAAAkBBhY
7AREUFjsBMjY1ETMyNicBJiIB3f7KCw4SxxAMqgwQxhIPC/7FCBgD3/6tDyH9wAwQEAwCQCEPAVM
JAAEAAAAAA+oDTAA0AAABMx8BAR8DDwMBDwMjLwwhLwE1NzUnPwEhPwQ1PwQCXgIFCQFxBAIEAgE
DBAf+ogYKBQUEAwQDAwICAQIBAQYJCf3mAgEDAgEBAh4KCAQCAQICAgIDA0wBBf7VAwQJCQkJCQf
+4QQGAgEBAQIDBAQFC50DBAQDAQICCuANAgECBQIDAqcMBQQDAQAAAQAAAAADTAPoABYAAAEGFRE
jIgYXARYyNwE2JisBETQmKwEiAYsIxhIPDAE5CRgJATUMDhPGEAyqDAPgCAz9wCEP/q0JCQFTDyE
CQAwQAAAAABIA3gABAAAAAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAADAAc
ADwABAAAAAAAEAAcAFgABAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAA
LABIAWwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAE
ECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyB
lLWljb25zUmVndWxhcmUtaWNvbnNlLWljb25zVmVyc2lvbiAxLjBlLWljb25zRm9udCBnZW5lcmF
0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0
AaQBjAG8AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGkAYwBvAG4AcwBlAC0AaQBjAG8AbgBzAFYAZQB
yAHMAaQBvAG4AIAAxAC4AMABlAC0AaQBjAG8AbgBzAEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQB
kAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQBAgEDAQQBBQADVXAxC2Fycm93LXJpZ2h0CURvd25fU29ydAA
AAA==) format('truetype');
  font-weight: normal;
  font-style: normal;
}
.pv-icons {
  font-family: 'e-pivot';
  font-style: normal;
  font-variant: normal;
  font-weight: normal;
  text-transform: none;
  line-height: 1;
}
.sb-icon-profit::before {
  content: '\e234';
  padding-left: 30px;
  margin: auto !important;
  color: #219122;
  size: 20px;
}
.sb-icon-neutral::before {
  content: '\e84d';
  padding-left: 30px;
  margin: auto !important;
```

```
  color: #82b5e9;
}
.sb-icon-loss::before {
  content: '\e239';
  padding-left: 30px;
  margin: auto !important;
  color: #ff2222;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/renewableEnergy-cs3" %}

Events

QueryCellInfo

The event queryCellInfo triggers while rendering row and value cells in the pivot table. It allows the user to customize the element of the current cell.

It has the following parameters:

- cell - it holds the current cell info
- data - it holds entire row data
- column - it holds column information for the current cell
- pivotview - it holds pivot instance

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        columnWidth: 140,
        queryCellInfo: function(args){
            //triggers for every value cell while rendering
        }
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs223" %}

HeaderCellInfo

The event headerCellInfo triggers while rendering the header cells in the pivot table. It allows the user to customize the element of the current header cell.

It has the following parameters:

- node - it holds the current header cell info

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
        columnWidth: 140,
        headerCellInfo: function(args){
```

```
            //triggers for every header cell while rendering
        }
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs224" %}

ColumnRender

The event columnRender triggers while framing each columns for rendering in the pivot table. It allows the user to customize the text alignment, column visibility, autofit, re-ordering, minimum and maximum width for a specific column. It has the following parameters:

- columns - It holds the leaf level columns (i.e., value headers) information.
- dataSourceSettings - It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- name - It holds the name of the event.
- stackedColumns - It holds the drilled columns (i.e., including column and value headers) information.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        filters: []
      },
      height: 350,
```

```
      gridSettings: {
        columnRender: function(args){
          // Here you can customize the specific columns.
          for (var i = 0; i < args.columns.length; i++) {
            args.columns[i].autoFit = true;
            args.columns[i].textAlign = "Right";
          }
        }
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs225" %}

CellClick

The event cellClick triggers while clicking a cell in the pivot table. For instance, using this event end-user can either add or remove styles, edit value and also perform any other DOM manipulations. It has the following parameters:

- currentCell - It holds the current cell information.
- data - It holds the clicked cell's data like axis, formatted text, actual text, row header, column header and value informations.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :cellClick="cellClick"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
```

```
      },
      height: 350
    }
  }
  methods: {
    cellClick:function(args: any) {
        //triggers on every cell click in pivot table
        args.currentCell.setAttribute("style", "background-color: red;")
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs226" %}

Summary customization in Vue Pivotview component

Show or hide grand totals

Allows to show or hide grand totals in rows and columns using the showGrandTotals property. To hide the grand totals in rows and columns, set the property showGrandTotals in dataSourceSettings to **false**.

End user can also hide grand totals for row or columns separately by setting the property showRowGrandTotals or showColumnGrandTotals in dataSourceSettings to **false** respectively.

By default, showGrandTotals, showRowGrandTotals and showColumnGrandTotals properties in dataSourceSettings are set as **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
```

```
        showGrandTotals: false
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs236" %}

Show grand totals at top or bottom

Allows to show grand totals either at top or bottom in rows and columns using the grandTotalsPosition property. To show the grand totals at top in rows and columns, set the grandTotalsPosition property in dataSourceSettings to **Top**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        grandTotalsPosition: 'Top',
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs237" %}

Show or hide sub-totals

Allows to show or hide sub-totals in rows and columns using the showSubTotals property. To hide all the sub-totals in rows and columns, set the property showSubTotals in dataSourceSettings to **false**. End user can also hide sub-totals for rows or columns separately by setting the property showRowSubTotals or showColumnSubTotals in dataSourceSettings to **false** respectively.

By default, showSubTotals, showRowSubTotals and showColumnSubTotals properties in dataSourceSettings are set as **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        showSubTotals: false
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs238" %}

Copyright © 2001 -2024 Syncfusion Inc.

Show or hide sub-totals for specific fields

Allows to show or hide sub-totals for specific fields in rows and columns using the showSubTotals property. To hide sub-totals for a specific field in row or column axis, set the property showSubTotals in rows or columns to **false** respectively.

By default, showSubTotals property in rows or columns is set as **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year', showSubTotals:
false }, { name: 'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country', showSubTotal: false }, { name: 'Products'
}],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs239" %}

Show sub-totals at top or bottom

Allows to show sub-totals either at top or bottom of the header group in rows and columns by using the subTotalsPosition property. By default, subTotalsPosition property is set to **Auto**, which means that

Copyright © 2001 -2024 Syncfusion Inc.

column sub-totals are displayed at the bottom and row sub-totals are displayed at the top of the header group in the pivot table.

To show sub-totals at top of the header group in rows and columns, set the subTotalsPosition property in dataSourceSettings to **Top**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }, { name:
'Year', items: ['FY 2015'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        subTotalsPosition: 'Top'
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs240" %}

To show sub-totals at bottom of the header group in rows and columns, set the subTotalsPosition property in dataSourceSettings to **Bottom**.

APP.VUE

```
<template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }, { name:
'Year', items: ['FY 2015'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        subTotalsPosition: 'Bottom'
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs241" %}

Show or hide totals using toolbar

It can also be achieved using built-in toolbar options by setting the showToolbar property in pivot table to **true**. Also, include the items **GrandTotal** and **SubTotal** within the toolbar property in pivot table. End user can now see "Show/Hide Grand totals" and "Show/Hide Sub totals" icons in toolbar UI automatically.

The grand totals and sub-totals can be dynamically displayed at the top or bottom of the pivot table's row and column axes by using the built-in options "Grand totals position" and "Subtotals position" available in the grand totals and sub-totals drop down menus, respectively.

APP.VUE

```
<template>
    <div id="app">
```

```
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:height="height" :showToolbar="showToolbar" :toolbar="toolbar" > </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import {
  PivotViewPlugin,
  IDataSet,
  Toolbar
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: { columnWidth: 140 },
      showToolbar: true,
      toolbar: [
        "SubTotal",
        "GrandTotal",
      ],
    };
  }
  provide: {
    pivotview: [
      Toolbar
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs242" %}

Hyper link in Vue Pivotview component

The pivot table allows you to show hyperlink option to the link data for individual cells that display in the pivot table. Also, the hyperlink can be enabled separately for row header, column header, value, and

Copyright © 2001 -2024 Syncfusion Inc.

summary cells using the hyperlinkSettings. It can be configured through code behind, during initial rendering. The settings available to show hyperlink to the cells are:

- showHyperlink: It allows to set the visibility of hyperlink in all cells.
- showRowHeaderHyperlink: It allows to set the visibility of hyperlink in row headers.
- showColumnHeaderHyperlink: It allows to set the visibility of hyperlink in column headers.
- showValueCellHyperlink: It allows to set the visibility of hyperlink in value cells.
- showSummaryCellHyperlink: It allows to set the visibility of hyperlink in summary cells.
- headerText: It allows to set the visibility of hyperlink based on header text.
- conditionalSettings: It allows to set the visibility of hyperlink based on specific condition.

By default, the hyperlink options are disabled for all cells in the pivot table.

Hyperlink for all cells

The pivot table has an option to show hyperlink option to all the cells that are currently displaying. To show hyperlink option, you need to set showHyperlink to **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"
:hyperlinkSettings="hyperlinkSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      hyperlinkSettings: {
          showHyperlink: true,
          cssClass: 'e-custom-class'
      }
    }
  }
```

```
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
.e-custom-class {
  color: #008cff;
  text-decoration: underline;
}
.e-custom-class:hover {
  color: red;
  text-decoration: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs143" %}

Hyperlink for row headers

The pivot table has an option to show hyperlink option to row header cells that are currently displaying. To show hyperlink option for row headers alone, you need to set showRowHeaderHyperlink to **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"
:hyperlinkSettings="hyperlinkSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      hyperlinkSettings: {
          showRowHeaderHyperlink: true,
          cssClass: 'e-custom-class'
      }
    }
```

```
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
.e-custom-class {
  color: #008cff;
  text-decoration: underline;
}
.e-custom-class:hover {
  color: red;
  text-decoration: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs144" %}

Hyperlink for column headers

The pivot table has an option to show hyperlink option to columns header cells that are currently displaying. To show hyperlink option for column headers alone, you need to set `showColumnHeaderHyperlink` to **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"
:hyperlinkSettings="hyperlinkSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      hyperlinkSettings: {
          showColumnHeaderHyperlink: true,
          cssClass: 'e-custom-class'
```

```
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
.e-custom-class {
  color: #008cff;
  text-decoration: underline;
}
.e-custom-class:hover {
  color: red;
  text-decoration: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs145" %}

Hyperlink for value cells

The pivot table has an option to show hyperlink option to value cells that are currently displaying. To show hyperlink option for values alone, you need to set `showValueCellHyperlink` to **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"
:hyperlinkSettings="hyperlinkSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      hyperlinkSettings: {
```

```
            showValueCellHyperlink: true,
            cssClass: 'e-custom-class'
        }
      }
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
.e-custom-class {
  color: #008cff;
  text-decoration: underline;
}
.e-custom-class:hover {
  color: red;
  text-decoration: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs146" %}

Hyperlink for summary cells

The pivot table has an option to show hyperlink option to summary value cells that are currently displaying. To show hyperlink option for summary values alone, you need to set showSummaryCellHyperlink to **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"
:hyperlinkSettings="hyperlinkSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      },
      height: 350,
      hyperlinkSettings: {
          showSummaryCellHyperlink: true,
          cssClass: 'e-custom-class'
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
.e-custom-class {
  color: #008cff;
  text-decoration: underline;
}
.e-custom-class:hover {
  color: red;
  text-decoration: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs147" %}

Condition based hyperlink

The pivot table has an option to show hyperlink option to the cells based on specific conditions. It can be configured using the conditionalSettings option through code behind, during initial rendering. The settings required to sort are:

- measure: Specifies the value field name to get visibility of hyperlink option for specific measure.
- conditions: Specifies the operator type such as equals, greater than, less than, etc.
- value1: Specifies the start value.
- value2: Specifies the end value.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"
:hyperlinkSettings="hyperlinkSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
```

```
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      hyperlinkSettings: {
        cssClass: 'e-custom-class',
        conditionalSettings: [{
            measure: 'Sold',
            conditions: 'Between',
            value1: 150,
            value2: 200
        },{
            label: 'Germany',
            conditions: 'GreaterThan',
            value1: 500
        }]
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
.e-custom-class {
  color: #008cff;
  text-decoration: underline;
}
.e-custom-class:hover {
  color: red;
  text-decoration: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs148" %}

Header based hyperlink

The pivot table has an option to show hyperlink option to the cells based on specific row or column. It can be configured using the `headerText` option through code behind, during initial rendering.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"
:hyperlinkSettings="hyperlinkSettings"> </ejs-pivotview>
    </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      hyperlinkSettings: {
        headerText: 'FY 2015.Q1.Units Sold',
        cssClass: 'e-custom-class'
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
.e-custom-class {
  color: #008cff;
  text-decoration: underline;
}
.e-custom-class:hover {
  color: red;
  text-decoration: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs149" %}

Event

The event hyperlinkCellClick fires on every hyperlink cell click.

It has following parameters - Cancel and CurrentCell. The parameter CurrentCell is used to customize the host cell element by any means. Meanwhile, when the parameter Cancel is set to **true**, applied customization will not be updated to the host cell element.

APP.VUE

```
<template>
    <div id="app">
```

```
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings"
:hyperlinkSettings="hyperlinkSettings"
:hyperlinkCellClick="hyperlinkCellClick"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      hyperlinkSettings: {
          showHyperlink: true,
          cssClass: 'e-custom-class'
      }
    }
  },
  methods: {
    hyperlinkCellClick: function(args) {
      args.cancel = false;
      args.currentCell.setAttribute("data-url",
"https://ej2.syncfusion.com/");//here we have redirected to EJ2 Syncfusion
on hyperlinkcell click
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
.e-custom-class {
  color: #008cff;
  text-decoration: underline;
}
.e-custom-class:hover {
  color: red;
  text-decoration: none;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs150" %}

See Also

- Apply condition based hyperlink for specific row or column

Tool bar in Vue Pivotview component

Toolbar option allows to access the frequently used features like switching between pivot table and pivot chart, changing chart types, conditional formatting, exporting, etc... with ease at runtime. This option can be enabled by setting the showToolbar property in pivot table to **true**. The toolbar property in pivot table accepts the collection of built-in toolbar options.

Built-in Toolbar Options

The following table shows built-in toolbar options and its actions.

Built-in Toolbar Options	Actions
New	Creates a new report
Save	Saves the current report
Save As	Save as current report
Rename	Renames the current report
Delete	Deletes the current report
Load	Loads any report from the report list
Grid	Shows pivot table
Chart	Shows a chart in any type from the built-in list and and option to enable/disable multiple axes
Exporting	Exports the pivot table as PDF/Excel/CSV and the pivot chart as PDF and image
Sub-total	Shows or hides sub totals
Grand Total	Shows or hides grand totals
Conditional Formatting	Shows the conditional formatting pop-up to apply formatting
Number Formatting	Shows the number formatting pop-up to apply number formatting
Field List	Shows the fieldlist pop-up
MDX	Shows the MDX query that was run to retrieve data from the OLAP data source. **NOTE: This applies only to the OLAP data source.**

The order of toolbar options can be changed by simply moving the position of items in the **ToolbarItems** collection. Also if end user wants to remove any toolbar option from getting displayed, it can be simply ignored from adding into the **ToolbarItems** collection.

APP.VUE

```
<template>
    <div id="app">
```

```
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:height="height"  :allowExcelExport="allowExcelExport"
:allowConditionalFormatting="allowConditionalFormatting"
:allowPdfExport="allowPdfExport" :showToolbar="showToolbar"
:allowNumberFormatting="allowNumberFormatting"
:allowCalculatedField="allowCalculatedField" :showFieldList="showFieldList"
:toolbar="toolbar" :saveReport="saveReport" :loadReport="loadReport"
:fetchReport="fetchReport" :renameReport="renameReport"
:removeReport="removeReport" :newReport="newReport"
:displayOption="displayOption"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import {
  PivotViewPlugin,
  GroupingBar,
  FieldList,
  IDataSet,
  CalculatedField,
  Toolbar,
  PDFExport,
  ExcelExport,
  ConditionalFormatting,
  NumberFormatting
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: { columnWidth: 140 },
      allowExcelExport: true,
      allowConditionalFormatting: true,
      allowNumberFormatting: true,
      allowPdfExport: true,
      displayOption: { view:'Both' },
      showToolbar: true,
      allowCalculatedField: true,
      showFieldList: true,
      toolbar: [
        "New",
        "Save",
        "SaveAs",
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        "Rename",
        "Remove",
        "Load",
        "Grid",
        "Chart",
        "Export",
        "SubTotal",
        "GrandTotal",
        "ConditionalFormatting",
        "NumberFormatting",
        "FieldList"
      ]
    };
  },
  methods: {
    saveReport: function(args: any) {
      let reports = [];
      let isSaved = false;
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reports = JSON.parse(localStorage.pivotviewReports);
      }
      if (args.report && args.reportName && args.reportName !== "") {
        reports.map(function(item: any) {
          if (args.reportName === item.reportName) {
            item.report = args.report;
            isSaved = true;
          }
        });
        if (!isSaved) {
          reports.push(args);
        }
        localStorage.pivotviewReports = JSON.stringify(reports);
      }
    },
    fetchReport: function(args: any) {
      let reportCollection = [];
      let reeportList: any = [];
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
      }
      reportCollection.map(function(item: any) {
        reeportList.push(item.reportName);
      });
      args.reportName = reeportList;
    },
    loadReport: function(args: any) {
      let pivotObj = document.getElementById('pivotview').ej2_instances[0];
      let reportCollection = [];
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
```

```
      ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
      }
      reportCollection.map(function(item: any) {
        if (args.reportName === item.reportName) {
          args.report = item.report;
        }
      });
      if (args.report) {
        pivotObj.dataSourceSettings =
JSON.parse(args.report).dataSourceSettings;
      }
    },
    removeReport: function(args: any) {
      let reportCollection = [];
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
      }
      for (let i = 0; i < reportCollection.length; i++) {
        if (reportCollection[i].reportName === args.reportName) {
          reportCollection.splice(i, 1);
        }
      }
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        localStorage.pivotviewReports = JSON.stringify(reportCollection);
      }
    },
    renameReport: function(args: any) {
      let reportCollection = [];
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
      }
      reportCollection.map(function(item: any) {
        if (args.reportName === item.reportName) {
          item.reportName = args.rename;
        }
      });
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        localStorage.pivotviewReports = JSON.stringify(reportCollection);
      }
    },
    newReport: function() {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.setProperties(
```

```
        {
          dataSourceSettings: {
            columns: [],
            rows: [],
            values: [],
            filters: []
          }
        },
        false
      );
    },
  },
  provide: {
    pivotview: [
      FieldList,
      CalculatedField,
      Toolbar,
      PDFExport,
      ExcelExport,
      ConditionalFormatting,
      NumberFormatting
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs245" %}

Show desired chart types in the dropdown menu

By default, all chart types are displayed in the dropdown menu included in the toolbar. However, based on the request for an application, we may need to show selective chart types on our own. This can be achieved using the chartTypes property. To know more about supporting chart types, click here.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings"  :height="height"
:showToolbar="showToolbar"  :toolbar="toolbar"
:displayOption="displayOption" :chartTypes="chartTypes"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import {
  PivotViewPlugin,
  IDataSet,
  Toolbar,
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

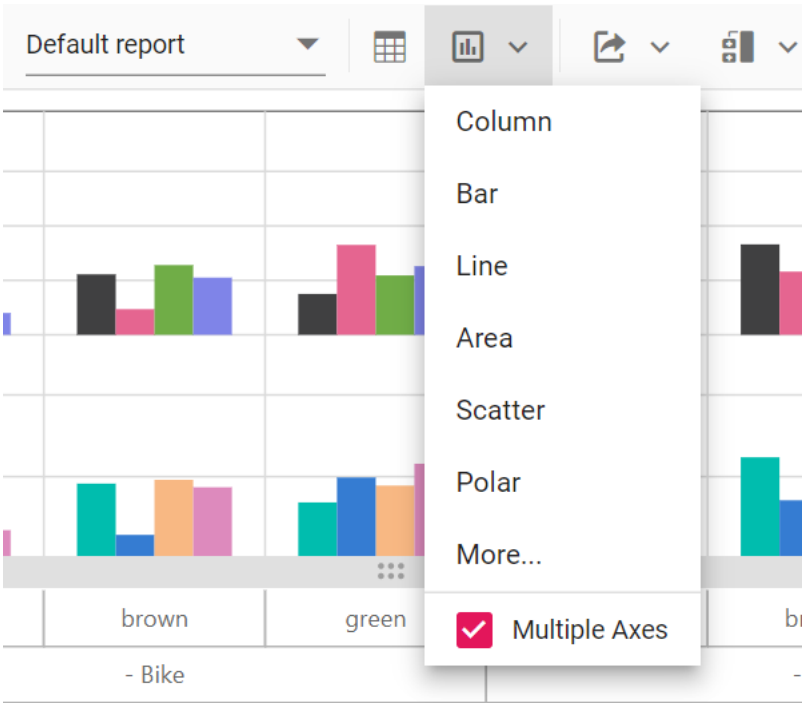
```
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      displayOption: { view:'Both' },
      showToolbar: true,
      chartTypes:['Column', 'Bar', 'Line', 'Area']
      toolbar: ['Grid', 'Chart']
    };
  },
  provide: {
    pivotview: [
      Toolbar,
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs246" %}

Switch the chart to multiple axes

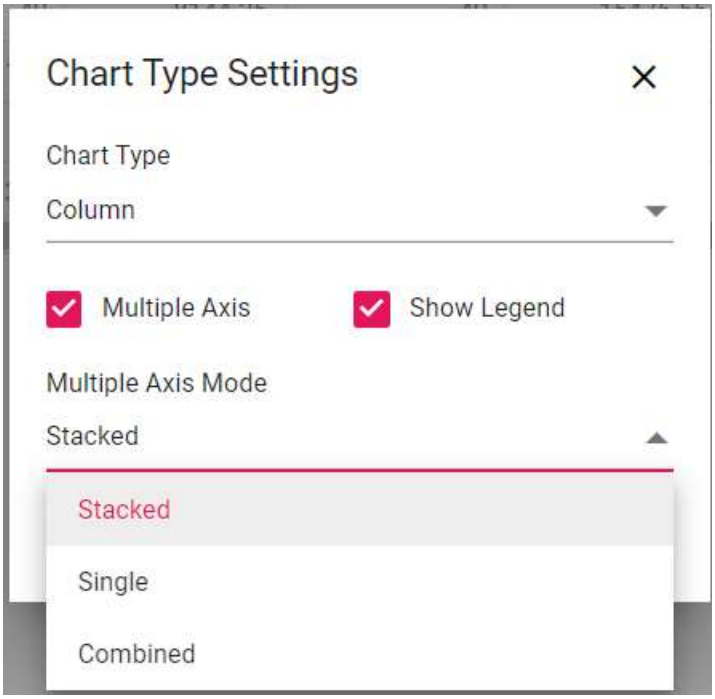
In the chart, the user can switch from single axis to multiple axes with the help of the built-in checkbox available inside the chart type dropdown menu in the toolbar. For more information refer here.

Copyright © 2001 -2024 Syncfusion Inc.



<!-- markdownlint-disable MD009 -->

There are three modes available in **Multiple Axis** option: Stacked, Single and Combined. The modes can be changed using “Multiple Axis Mode” drop-down list which appears while clicking the **More...** option.

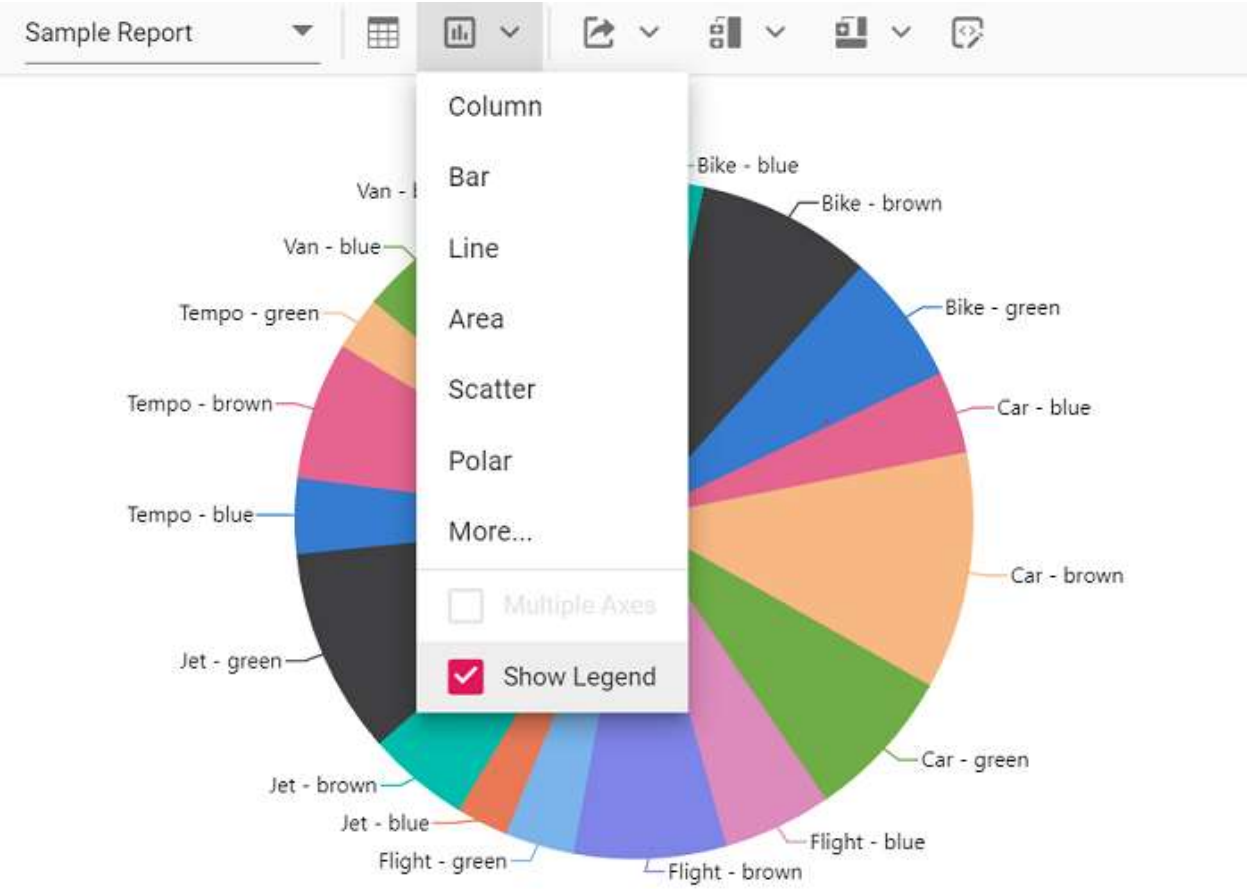


Show or hide legend

In the chart, legend can be shown or hidden dynamically with the help of the built-in option available in the chart type drop-down menu.

Copyright © 2001 -2024 Syncfusion Inc.

By default, the legend is not be visible for the accumulation chart types like pie, doughnut, pyramid, and funnel. Users can enable or disable using the built-in checkbox option.



Adding custom option to the toolbar

In addition to the existing built-in toolbar items, new toolbar item(s) may also be included. This can be achieved by using the toolbarRender event. The action of the new toolbar item(s) can also be defined within this event.

The new toolbar item(s) can be added to the desired position in the toolbar using the splice option.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:height="height" :showToolbar="showToolbar" :toolbar="toolbar"
:toolbarRender="beforeToolbarRender" :displayOption="displayOption"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import {
```

```
  PivotViewPlugin,
  IDataSet,
  Toolbar,
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: { columnWidth: 140 },
      displayOption: { view:'Both' },
      showToolbar: true,
      toolbar: [
        "Expand/Collpase",
      ]
    };
  },
  methods: {
    beforeToolbarRender: function (args: any) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      args.customToolbar.splice(12, 0, {
            prefixIcon: 'e-pivotview-expand e-icons', tooltipText:
'Expand/Collapse',
            click: this.toolbarClicked.bind(this),
        });
    },
    toolbarClicked: function (args: any) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.dataSourceSettings.expandAll =
!pivotGridObj.dataSourceSettings.expandAll;
    }
  },
  provide: {
    pivotview: [
      Toolbar,
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs247" %}

In the above topic, we have seen how to add an icon as one of the toolbar item in toolbar panel. In the next topic, we are going to see how to frame the entire toolbar panel and how to add a custom control in it.

Toolbar Template

It allows to customize the toolbar panel by using template option. It allows any custom control to be used as one of the toolbar item inside the toolbar panel. It can be achieved by two ways,

Here, the entire toolbar panel can be framed in HTML elements that are appended at the top of the pivot table. The **id** of the HTML element needs to be set in the toolbarTemplate property in-order to map it to the pivot table.

APP.VUE

```
<template>
    <div id="app">
    <div id="template" >
    <div>
     <ejs-button id="expandall" :isPrimary="isPrimary" cssClass='e-flat' v-
on:click.native="expandAll">EXPAND ALL</ejs-button>
      <ejs-button id="collapseall" :isPrimary="isPrimary" cssClass='e-flat'
v-on:click.native="collapseAll">COLLAPSE ALL</ejs-button>
      </div>
       </div>
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:height="height" :showToolbar="showToolbar"
:toolbarTemplate="toolbarTemplate"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { ButtonPlugin, ChangeEventArgs } from '@syncfusion/ej2-vue-buttons';
import {
  PivotViewPlugin,
  IDataSet,
  Toolbar,
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
```

```
      gridSettings: { columnWidth: 140 },
      showToolbar: true,
      toolbarTemplate: '#template',
      isPrimary:true
    };
  },
  methods: {
    expandAll: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.dataSourceSettings.expandAll=true;
    },
    collapseAll: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.dataSourceSettings.expandAll=false;
    }
  },
  provide: {
    pivotview: [
      Toolbar,
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs248" %}

Another option allows to frame a custom toolbar item using HTML elements and include in the toolbar panel at the desired position. The custom toolbar items can be declared as control **instance** or element **id** in the toolbar property in pivot table.

APP.VUE

```
<template>
    <div id="app">
     <ejs-button id="enablertl" :isPrimary="isPrimary" cssClass='e-flat' v-
on:click.native="enableRtl">ENABLE RTL</ejs-button>
      <ejs-button id="disablertl" :isPrimary="isPrimary" cssClass='e-flat'
v-on:click.native="disableRtl">DISABLE RTL</ejs-button>
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:height="height" :showToolbar="showToolbar" :toolbar="toolbar"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { ButtonPlugin, ChangeEventArgs } from '@syncfusion/ej2-vue-buttons';
import {
  PivotViewPlugin,
  IDataSet,
  Toolbar,
```

```
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: { columnWidth: 140 },
      showToolbar: true,
      toolbar: [{template:'#enablertl'}, {template:'#disablertl'}],
      isPrimary:true
    };
  },
  methods: {
    enableRtl: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.enableRtl=true;
    },
    disableRtl: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.enableRtl=false;
    }
  },
  provide: {
    pivotview: [
      Toolbar,
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs249" %}

Note: For both options, the actions for the toolbar template items can be defined in the event toolbarClick. Also, if the toolbar item is a custom control then its built-in events can also be accessed.

<!-- markdownlint-disable MD009 -->

Copyright © 2001 -2024 Syncfusion Inc.

Save and load report as a JSON file

The current pivot report can be saved as a JSON file in the desired path and loaded back to the pivot table at any time.

APP.VUE

```
<template>
    <div id="app">
     <div>
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height= "height"
:showFieldList="showFieldList" :dataBound="ondataBound"></ejs-pivotview>
      </div>
    <a id="save" class="btn btn-primary">Save</a><div class="fileUpload btn
btn-primary"><span>Load</span><input id="files" type="file" class="upload"
/></div>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
   methods: {
    ondataBound: function (args: any) {
        let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
        var dataSource =
JSON.parse(pivotGridObj.getPersistData()).dataSourceSettings.dataSource;
        var a = document.getElementById('save');
        var mime_type = 'application/octet-stream'; // text/html, image/png,
et c
        a.setAttribute('download', 'pivot.JSON');
        a.href = 'data:'+ mime_type +'base64,'+
btoa(JSON.stringify(dataSource) || '');
        document.getElementById('files').addEventListener('change',
this.readBlob, false);
    },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    readBlob: function (args: any) {
        var files = document.getElementById('load').files;
        var file = files[0];
        var start = 0;
        var stop = file.size - 1;
        var reader = new FileReader();
        reader.onloadend = function(evt) {
            if (evt.target.readyState == FileReader.DONE) {
              let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
              pivotGridObj.dataSource = JSON.parse(evt.target.result);
            }
        };
        var blob = file.slice(start, stop + 1);
        reader.readAsBinaryString(blob);
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
@import
"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css";
.fileUpload {
    position: relative;
    overflow: hidden;
    margin: 10px;
}
.fileUpload input.upload {
    position: absolute;
    top: 0;
    right: 0;
    margin: 0;
    padding: 0;
    font-size: 20px;
    cursor: pointer;
    opacity: 0;
    filter: alpha(opacity=0);
}
</style>
```

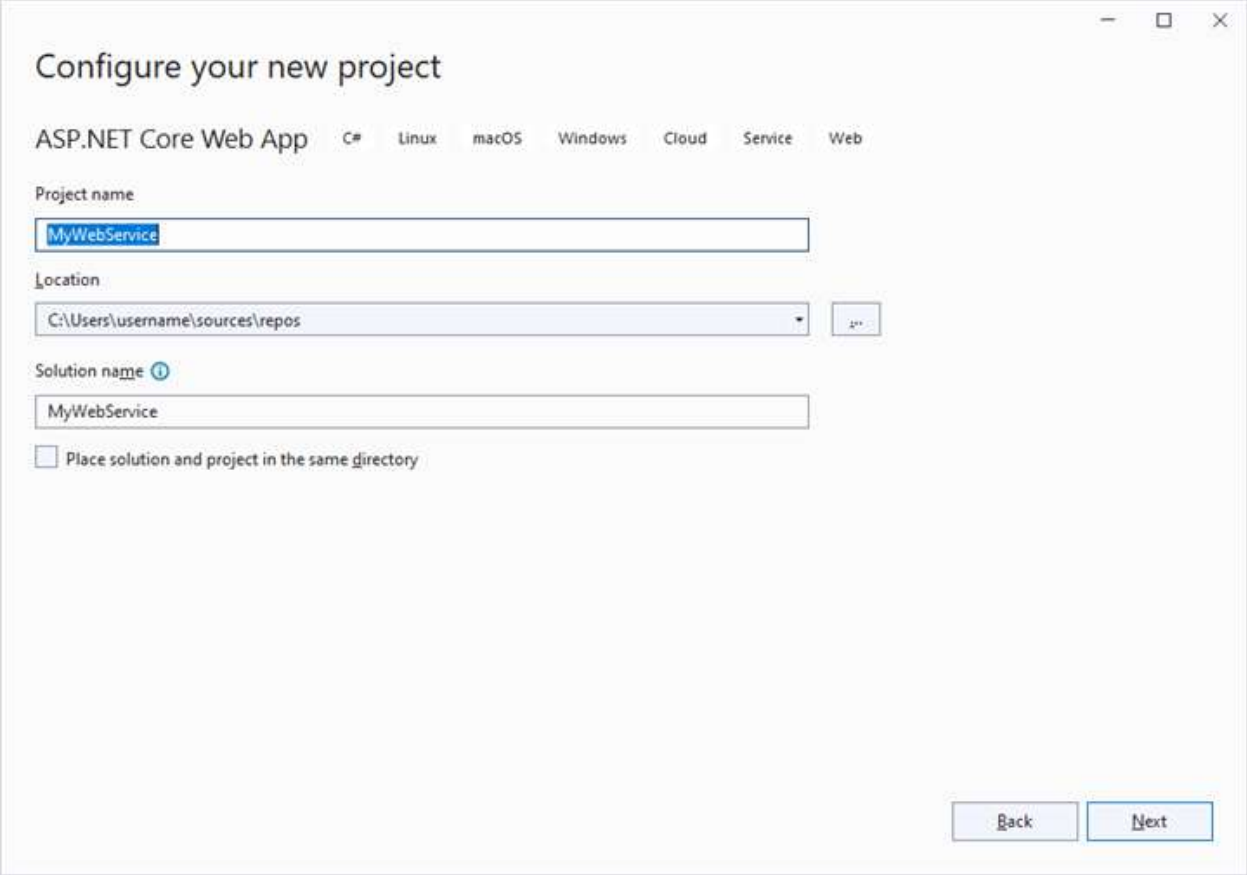
{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs250" %}

Save and load reports to a SQL database

SQL Server is a relational database management system (RDBMS) that can be used to store and manage large amounts of data. In this topic, we will see how to save, save as, rename, load, delete, and add reports between a SQL Server database and a Vue Pivot Table at runtime.

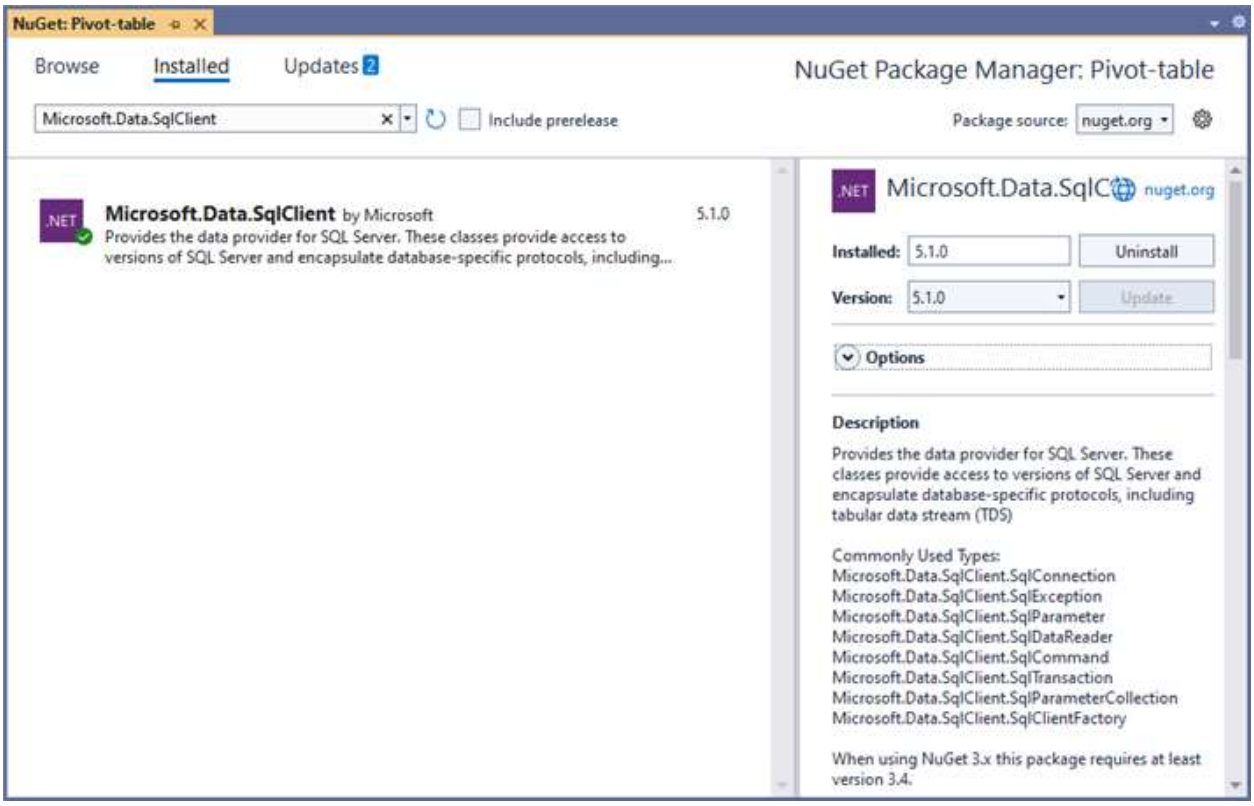
Create a Web API service to connect to a SQL Server database

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document link.



2. To connect a SQL Server database using the Microsoft SqlClient in our application, we need to install the Microsoft.Data.SqlClient NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **Microsoft.Data.SqlClient** and install it.

Copyright © 2001 -2024 Syncfusion Inc.



3. Under the **Controllers** folder, create a Web API controller (aka, PivotController.cs) file that aids in data communication with the Pivot Table.

4. In the Web API Controller (aka, PivotController), the **OpenConnection** method is used to connect to the SQL database. The **GetDataTable** method then processes the specified SQL query string, retrieves data from the database, and converts it into a **DataTable** using **SqlCommand** and **SqlDataAdapter**. This **DataTable** can be used to retrieve saved reports and modify them further as shown in the code block below.

[PivotController.cs]

`csharp

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using System.Data;
namespace MyWebService.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpPost]
```

```
[Route("Pivot/SaveReport")]
public void SaveReport([FromBody] Dictionary<string, string> reportArgs)
{
SaveReportToDB(reportArgs["reportName"], reportArgs["report"]);
}
[HttpPost]
[Route("Pivot/FetchReport")]
public IActionResult FetchReport()
{
return Ok((FetchReportListFromDB()));
}
[HttpPost]
[Route("Pivot/LoadReport")]
public IActionResult LoadReport([FromBody] Dictionary<string, string> reportArgs)
{
return Ok((LoadReportFromDB(reportArgs["reportName"])));
}
[HttpPost]
[Route("Pivot/RemoveReport")]
public void RemoveReport([FromBody] Dictionary<string, string> reportArgs)
{
RemoveReportFromDB(reportArgs["reportName"]);
}
[HttpPost]
[Route("Pivot/RenameReport")]
public void RenameReport([FromBody] RenameReportDB reportArgs)
{
RenameReportInDB(reportArgs.ReportName, reportArgs.RenameReport, reportArgs.isReportExists);
}
public class RenameReportDB
{
public string ReportName { get; set; }
public string RenameReport { get; set; }
```

```
public bool isReportExists { get; set; }
}
private void SaveReportToDB(string reportName, string report)
{
SqlConnection sqlConn = OpenConnection();
bool isDuplicate = true;
SqlCommand cmd1 = null;
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
if ((row["ReportName"] as string).Equals(reportName))
{
isDuplicate = false;
cmd1 = new SqlCommand("update ReportTable set Report=@Report where ReportName like
@ReportName", sqlConn);
}
}
if (isDuplicate)
{
cmd1 = new SqlCommand("insert into ReportTable (ReportName,Report)
Values(@ReportName,@Report)", sqlConn);
}
cmd1.Parameters.AddWithValue("@ReportName", reportName);
cmd1.Parameters.AddWithValue("@Report", report.ToString());
cmd1.ExecuteNonQuery();
sqlConn.Close();
}
private string LoadReportFromDB(string reportName)
{
SqlConnection sqlConn = OpenConnection();
string report = string.Empty;
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
if ((row["ReportName"] as string).Equals(reportName))
{
```

```
report = (string)row["Report"];
break;
}
}
sqlConn.Close();
return report;
}
private List<string> FetchReportListFromDB()
{
SqlConnection sqlConn = OpenConnection();
List<string> reportNames = new List<string>();
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
if (!string.IsNullOrEmpty(row["ReportName"] as string))
{
reportNames.Add(row["ReportName"].ToString());
}
}
sqlConn.Close();
return reportNames;
}
private void RenameReportInDB(string reportName, string renameReport, bool isReportExists)
{
SqlConnection sqlConn = OpenConnection();
SqlCommand cmd1 = null;
if (isReportExists)
{
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
if ((row["ReportName"] as string).Equals(reportName))
{
cmd1 = new SqlCommand("delete from ReportTable where ReportName like '%" + reportName + "%'",
sqlConn);
```

```
break;
}
}
cmd1.ExecuteNonQuery();
}
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
if ((row["ReportName"] as string).Equals(reportName))
{
cmd1 = new SqlCommand("update ReportTable set ReportName=@RenameReport where ReportName like '%" + reportName + "%'", sqlConn);
break;
}
}
cmd1.Parameters.AddWithValue("@RenameReport", renameReport);
cmd1.ExecuteNonQuery();
sqlConn.Close();
}
private void RemoveReportFromDB(string reportName)
{
SqlConnection sqlConn = OpenConnection();
SqlCommand cmd1 = null;
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
if ((row["ReportName"] as string).Equals(reportName))
{
cmd1 = new SqlCommand("delete from ReportTable where ReportName like '%" + reportName + "%'", sqlConn);
break;
}
}
cmd1.ExecuteNonQuery();
sqlConn.Close();
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
private SqlConnection OpenConnection()
{
// Replace with your own connection string.
string connectionString = @"<Enter your valid connection string here>";
SqlConnection sqlConn = new SqlConnection(connectionString);
sqlConn.Open();
return sqlConn;
}
private DataTable GetDataTable(SqlConnection sqlConn)
{
string xquery = "select * from ReportTable";
SqlCommand cmd = new SqlCommand(xquery, sqlConn);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
return dt;
}
}
}
`
```

5. When you run the app, it will be hosted at https://localhost:44313. You can use the hosted URL to save and load reports in the SQL database from the Pivot Table.

Further, let us explore more on how to save, load, rename, delete, and add reports using the built-in toolbar options via Web API controller (aka, PivotController) one-by-one.

Saving a report

When you select the **“Save a report”** option from the toolbar, the saveReport event is triggered. In this event, an AJAX request is made to the Web API controller's **SaveReport** method, passing the name of the current report and the current report, which you can use to check and save in the SQL database.

For example, the report shown in the following code snippet will be passed to the **SaveReport** method along with the report name **“Sample Report”** and saved in the SQL database.

[App.vue]

```typescript
<template>
<div>
<div class="control-section" style="overflow: auto">
```

```
<div class="content-wrapper">
<ejs-pivotview id="pivotview" ref="pivotview" :dataSourceSettings="dataSourceSettings"
:width="width" :allowExcelExport="allowExcelExport"
:allowConditionalFormatting="allowConditionalFormatting"
:allowPdfExport="allowPdfExport" :showToolbar="showToolbar"
:allowCalculatedField="allowCalculatedField"
:showFieldList="showFieldList" :toolbar="toolbar" :saveReport="saveReport" :loadReport="loadReport"
:fetchReport="fetchReport" :renameReport="renameReport" :removeReport="removeReport"
:newReport="newReport"
:toolbarRender="beforeToolbarRender" :displayOption="displayOption"
:chartSettings="chartSettings"></ejs-pivotview>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import {
PivotViewPlugin, FieldList, CalculatedField, Toolbar, ConditionalFormatting, NumberFormatting
} from "@syncfusion/ej2-vue-pivotview";
import { enableRipple } from "@syncfusion/ej2-base";
enableRipple(false);
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
columns: [{ name: 'Year', caption: 'Production Year' }, { name: 'Quarter' }],
dataSource: data,
expandAll: false,
filters: [],
formatSettings: [{ name: 'Amount', format: 'C0' }],
rows: [{ name: 'Country' }, { name: 'Products' }],
values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount', caption: 'Sold Amount' }]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
displayOption: { view: "Both" },
chartSettings: {
value: "Amount",
enableExport: true,
chartSeries: { type: "Column", animation: { enable: false } },
enableMultipleAxis: false,
},
toolbar: [
"New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid", "Chart", "MDX", "Export", "SubTotal",
"GrandTotal",
"ConditionalFormatting", "FieldList"
],
allowExcelExport: true,
allowConditionalFormatting: true,
allowPdfExport: true,
showToolbar: true,
allowCalculatedField: true,
showFieldList: true,
width: "700",
};
},
methods: {
saveReport(args) {
var report = JSON.parse(args.report);
report.dataSourceSettings.dataSource = [];
fetch('https://localhost:44313/Pivot/SaveReport', {
method: 'POST',
headers: {
'Accept': 'application/json',
'Content-Type': 'application/json',
},
body: JSON.stringify({ reportName: args.reportName, report: JSON.stringify(report) })
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}).then(response => {
this.fetchReport(args);
});
}
},
provide: {
pivotview: [FieldList, CalculatedField, Toolbar, ConditionalFormatting, NumberFormatting]
}
}
`
```

[PivotController.cs]

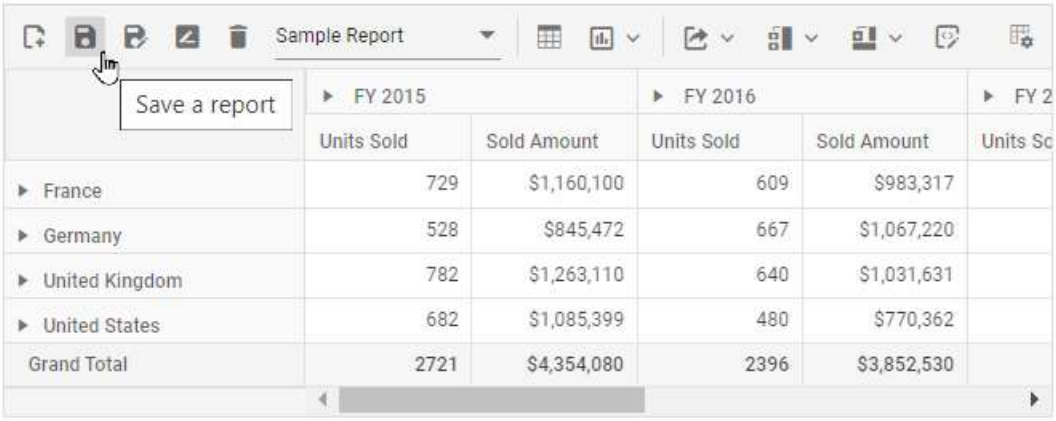
```csharp
namespace MyWebApp.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpPost]
[Route("Pivot/SaveReport")]
public void SaveReport([FromBody] Dictionary<string, string> reportArgs)
{
SaveReportToDB(reportArgs["reportName"], reportArgs["report"]);
}
private void SaveReportToDB(string reportName, string report)
{
SqlConnection sqlConn = OpenConnection();
bool isDuplicate = true;
SqlCommand cmd1 = null;
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
if ((row["ReportName"] as string).Equals(reportName))
{
```

Copyright © 2001 -2024 Syncfusion Inc.

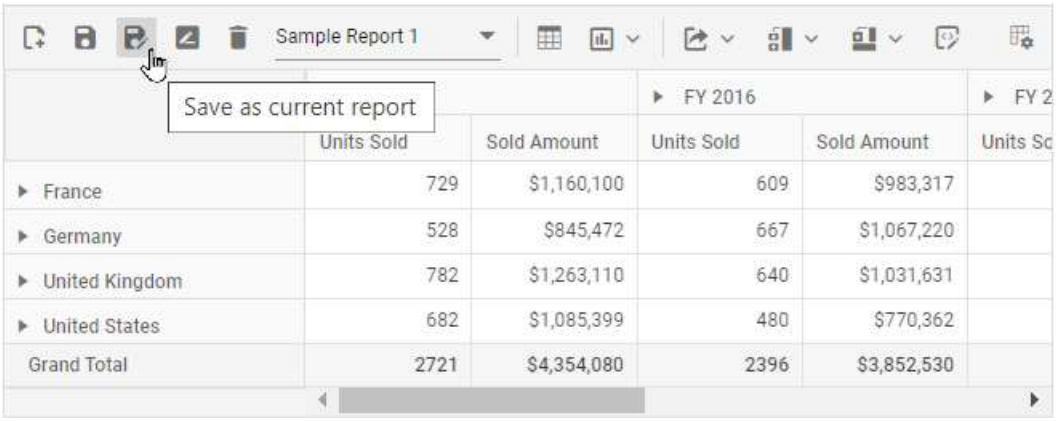
```
isDuplicate = false;
cmd1 = new SqlCommand("update ReportTable set Report=@Report where ReportName like
@ReportName", sqlConn);
}
}
if (isDuplicate)
{
cmd1 = new SqlCommand("insert into ReportTable (ReportName,Report)
Values(@ReportName,@Report)", sqlConn);
}
cmd1.Parameters.AddWithValue("@ReportName", reportName);
cmd1.Parameters.AddWithValue("@Report", report.ToString());
cmd1.ExecuteNonQuery();
sqlConn.Close();
}
private SqlConnection OpenConnection()
{
// Replace with your own connection string.
string connectionString = @"<Enter your valid connection string here>";
SqlConnection sqlConn = new SqlConnection(connectionString);
sqlConn.Open();
return sqlConn;
}
private DataTable GetDataTable(SqlConnection sqlConn)
{
string xquery = "select * from ReportTable";
SqlCommand cmd = new SqlCommand(xquery, sqlConn);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
return dt;
}
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

`



In the meantime, you can save a duplicate of the current report to the SQL Server database with a different name by selecting **"Save as current report"** from the toolbar. The saveReport event will then be triggered with the new report name **"Sample Report 1"** and the current report. You can save them to the SQL Server database after passing them to the Web API service, as mentioned above.



Loading a report

When you select the dropdown menu item from the toolbar, the loadReport event is triggered. In this event, an AJAX request is made to the **LoadReport** method of the Web API controller, passing the name of the selected report. The method uses this information to search for the report in the SQL database, fetch it, and load it into the pivot table.

For example, if the report name **"Sample Report 1"** is selected from a dropdown menu and passed, the **LoadReport** method will use that name to search for the report in the SQL database, retrieve it, and then load it into the pivot table.

[App.vue]

`typescript

<template>

Copyright © 2001 -2024 Syncfusion Inc.

```
<ejs-pivotview id="pivotview" ref="pivotview" :dataSourceSettings="dataSourceSettings"
:width="width" :allowExcelExport="allowExcelExport"
:allowConditionalFormatting="allowConditionalFormatting"
:allowPdfExport="allowPdfExport" :showToolbar="showToolbar"
:allowCalculatedField="allowCalculatedField"
:showFieldList="showFieldList" :toolbar="toolbar" :saveReport="saveReport" :loadReport="loadReport"
:fetchReport="fetchReport" :renameReport="renameReport" :removeReport="removeReport"
:newReport="newReport"
:toolbarRender="beforeToolbarRender" :displayOption="displayOption"
:chartSettings="chartSettings"></ejs-pivotview>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import {
PivotViewPlugin, FieldList, CalculatedField, Toolbar, ConditionalFormatting, NumberFormatting
} from "@syncfusion/ej2-vue-pivotview";
import { enableRipple } from "@syncfusion/ej2-base";
enableRipple(false);
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
columns: [{ name: 'Year', caption: 'Production Year' }, { name: 'Quarter' }],
dataSource: data,
expandAll: false,
filters: [],
formatSettings: [{ name: 'Amount', format: 'C0' }],
rows: [{ name: 'Country' }, { name: 'Products' }],
values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount', caption: 'Sold Amount' }]
},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
displayOption: { view: "Both" },
chartSettings: {
value: "Amount",
enableExport: true,
chartSeries: { type: "Column", animation: { enable: false } },
enableMultipleAxis: false,
},
toolbar: [
"New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid", "Chart", "MDX", "Export", "SubTotal", "GrandTotal",
"ConditionalFormatting", "FieldList"
],
allowExcelExport: true,
allowConditionalFormatting: true,
allowPdfExport: true,
showToolbar: true,
allowCalculatedField: true,
showFieldList: true,
width: "700",
};
},
methods: {
loadReport(args) {
fetch('https://localhost:44313/Pivot/LoadReport', {
method: 'POST',
headers: {
'Accept': 'application/json',
'Content-Type': 'application/json',
},
body: JSON.stringify({ reportName: args.reportName })
}).then(res => res.json())
.then(response => {
if (response) {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
var report = JSON.parse(response);
var pivotTableObj = (this.$refs.pivotview).ej2Instances;
report.dataSourceSettings.dataSource = pivotTableObj.dataSourceSettings.dataSource;
pivotTableObj.dataSourceSettings = report.dataSourceSettings;
}
});
}
},
provide: {
pivotview: [FieldList, CalculatedField, Toolbar, ConditionalFormatting, NumberFormatting]
}
}
`
```

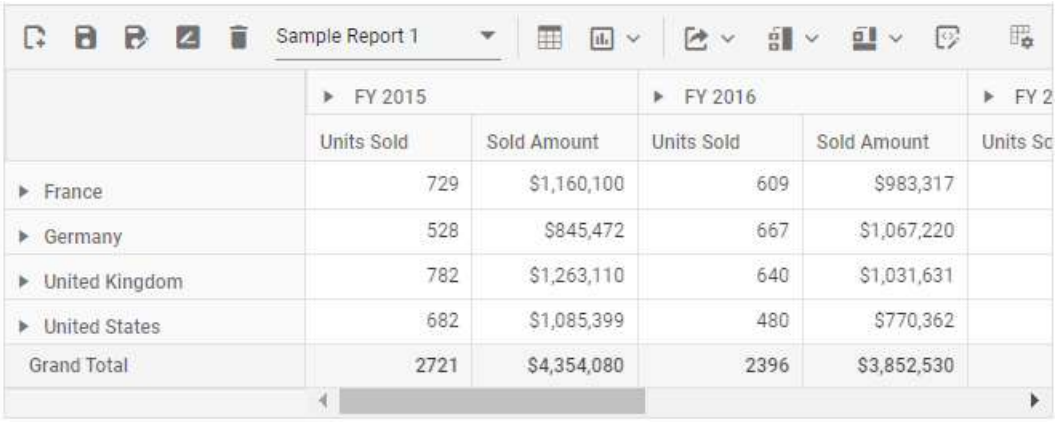
[PivotController.cs]

```csharp
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using System.Data;
namespace MyWebApp.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpPost]
[Route("Pivot/LoadReport")]
public IActionResult LoadReport([FromBody] Dictionary<string, string> reportArgs)
{
return Ok((LoadReportFromDB(reportArgs["reportName"])));
}
private string LoadReportFromDB(string reportName)
{
SqlConnection sqlConn = OpenConnection();
```

Copyright © 2001 -2024 Syncfusion Inc.

```
string report = string.Empty;
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
if ((row["ReportName"] as string).Equals(reportName))
{
report = (string)row["Report"];
break;
}
}
sqlConn.Close();
return report;
}
private SqlConnection OpenConnection()
{
// Replace with your own connection string.
string connectionString = @"<Enter your valid connection string here>";
SqlConnection sqlConn = new SqlConnection(connectionString);
sqlConn.Open();
return sqlConn;
}
private DataTable GetDataTable(SqlConnection sqlConn)
{
string xquery = "select * from ReportTable";
SqlCommand cmd = new SqlCommand(xquery, sqlConn);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
return dt;
}
}
}
`
```

Copyright © 2001 -2024 Syncfusion Inc.



Renaming a report

When you select the **“Rename a current report”** option from the toolbar, the renameReport event is triggered. In this event, an AJAX request is made to the **RenameReport** method of the Web API controller, passing the current and new report names, where you can use the current report name to identify the report and resave it with the new report name in the SQL database.

For example, if we rename the current report from **“Sample Report 1”** to **“Sample Report 2”**, both **“Sample Report 1”** and **“Sample Report 2”** will be passed to the **RenameReport** method, which will rename the current report with the new report name **“Sample Report 2”** in the SQL database.

[App.vue]

`typescript

<template>

<ejs-pivotview id="pivotview" ref="pivotview" :dataSourceSettings="dataSourceSettings"

:width="width" :allowExcelExport="allowExcelExport"

:allowConditionalFormatting="allowConditionalFormatting"

:allowPdfExport="allowPdfExport" :showToolbar="showToolbar"
:allowCalculatedField="allowCalculatedField"

:showFieldList="showFieldList" :toolbar="toolbar" :saveReport="saveReport" :loadReport="loadReport"

:fetchReport="fetchReport" :renameReport="renameReport" :removeReport="removeReport"
:newReport="newReport"

:toolbarRender="beforeToolbarRender" :displayOption="displayOption"

:chartSettings="chartSettings"></ejs-pivotview>

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import {
PivotViewPlugin, FieldList, CalculatedField, Toolbar, ConditionalFormatting, NumberFormatting
} from "@syncfusion/ej2-vue-pivotview";
import { enableRipple } from "@syncfusion/ej2-base";
enableRipple(false);
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
columns: [{ name: 'Year', caption: 'Production Year' }, { name: 'Quarter' }],
dataSource: data,
expandAll: false,
filters: [],
formatSettings: [{ name: 'Amount', format: 'C0' }],
rows: [{ name: 'Country' }, { name: 'Products' }],
values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount', caption: 'Sold Amount' }]
},
displayOption: { view: "Both" },
chartSettings: {
value: "Amount",
enableExport: true,
chartSeries: { type: "Column", animation: { enable: false } },
enableMultipleAxis: false,
},
toolbar: [
"New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid", "Chart", "MDX", "Export", "SubTotal",
"GrandTotal",
"ConditionalFormatting", "FieldList"
],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
allowExcelExport: true,
allowConditionalFormatting: true,
allowPdfExport: true,
showToolbar: true,
allowCalculatedField: true,
showFieldList: true,
width: "700",
};
},
methods: {
renameReport(args) {
fetch('https://localhost:44313/Pivot/RenameReport', {
method: 'POST',
headers: {
'Accept': 'application/json',
'Content-Type': 'application/json',
},
body: JSON.stringify({ reportName: args.reportName, renameReport: args.rename, isReportExists: args.isReportExists })
}).then(response => {
this.fetchReport(args);
});
}
},
provide: {
pivotview: [FieldList, CalculatedField, Toolbar, ConditionalFormatting, NumberFormatting]
}
}
`
```

[PivotController.cs]

```csharp
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
```

```
using System.Data;
namespace MyWebApp.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpPost]
[Route("Pivot/RenameReport")]
public void RenameReport([FromBody] RenameReportDB reportArgs)
{
RenameReportInDB(reportArgs.ReportName, reportArgs.RenameReport, reportArgs.isReportExists);
}
public class RenameReportDB
{
public string ReportName { get; set; }
public string RenameReport { get; set; }
public bool isReportExists { get; set; }
}
private void RenameReportInDB(string reportName, string renameReport, bool isReportExists)
{
SqlConnection sqlConn = OpenConnection();
SqlCommand cmd1 = null;
if (isReportExists)
{
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
if ((row["ReportName"] as string).Equals(reportName))
{
cmd1 = new SqlCommand("delete from ReportTable where ReportName like '%" + reportName + "%'",
sqlConn);
break;
}
```

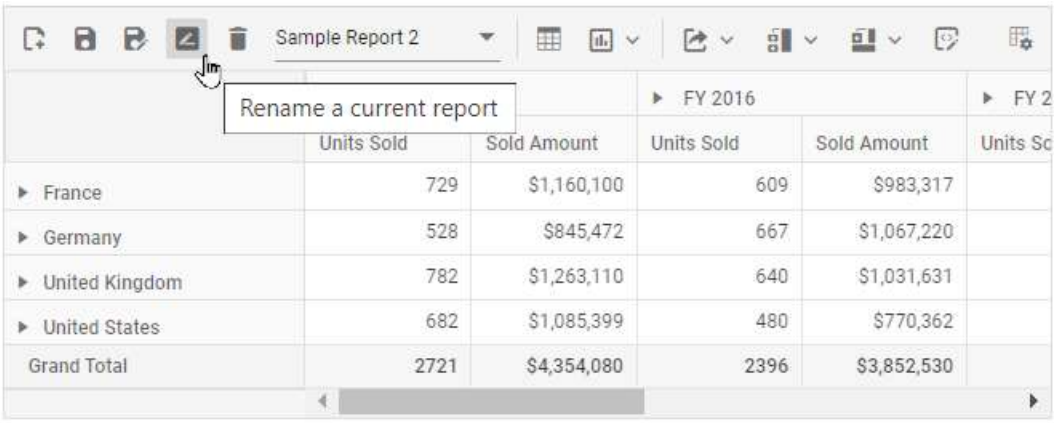
```
}
cmd1.ExecuteNonQuery();
}
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
if ((row["ReportName"] as string).Equals(reportName))
{
cmd1 = new SqlCommand("update ReportTable set ReportName=@RenameReport where ReportName like '%" + reportName + "%'", sqlConn);
break;
}
}
cmd1.Parameters.AddWithValue("@RenameReport", renameReport);
cmd1.ExecuteNonQuery();
sqlConn.Close();
}
private SqlConnection OpenConnection()
{
// Replace with your own connection string.
string connectionString = @"<Enter your valid connection string here>";
SqlConnection sqlConn = new SqlConnection(connectionString);
sqlConn.Open();
return sqlConn;
}
private DataTable GetDataTable(SqlConnection sqlConn)
{
string xquery = "select * from ReportTable";
SqlCommand cmd = new SqlCommand(xquery, sqlConn);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
return dt;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

}

}

`



Deleting a report

When you select the **“Delete a current report”** option from the toolbar, the removeReport event is triggered. In this event, an AJAX request is made to the **RemoveReport** method of the Web API controller, passing the current report name to identify and delete the appropriate report from the SQL database.

Note: * If the current report **n** from the pivot table is deleted, the pivot table will automatically load the last report from the report list.

* When a report is removed from a pivot table with only one report, the SQL database refreshes; however, the pivot table will continue to show the removed report until a new report is added to the pivot table.

For example, if we delete the current report **“Sample Report 2”** from the pivot table, the current report name **“Sample Report 2”** is passed to the **RemoveReport** method, which allows you to identify and delete the report from the SQL database.

[App.vue]

`typescript

<template>

<ejs-pivotview id="pivotview" ref="pivotview" :dataSourceSettings="dataSourceSettings"

:width="width" :allowExcelExport="allowExcelExport"

:allowConditionalFormatting="allowConditionalFormatting"

:allowPdfExport="allowPdfExport" :showToolbar="showToolbar"
:allowCalculatedField="allowCalculatedField"

:showFieldList="showFieldList" :toolbar="toolbar" :saveReport="saveReport" :loadReport="loadReport"

Copyright © 2001 -2024 Syncfusion Inc.

```
:fetchReport="fetchReport" :renameReport="renameReport" :removeReport="removeReport"
:newReport="newReport"
:toolbarRender="beforeToolbarRender" :displayOption="displayOption"
:chartSettings="chartSettings"></ejs-pivotview>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import {
PivotViewPlugin, FieldList, CalculatedField, Toolbar, ConditionalFormatting, NumberFormatting
} from "@syncfusion/ej2-vue-pivotview";
import { enableRipple } from "@syncfusion/ej2-base";
enableRipple(false);
Vue.use(PivotViewPlugin);
export default {
data() {
return {
dataSourceSettings: {
columns: [{ name: 'Year', caption: 'Production Year' }, { name: 'Quarter' }],
dataSource: data,
expandAll: false,
filters: [],
formatSettings: [{ name: 'Amount', format: 'C0' }],
rows: [{ name: 'Country' }, { name: 'Products' }],
values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount', caption: 'Sold Amount' }]
},
displayOption: { view: "Both" },
chartSettings: {
value: "Amount",
enableExport: true,
chartSeries: { type: "Column", animation: { enable: false } },
```

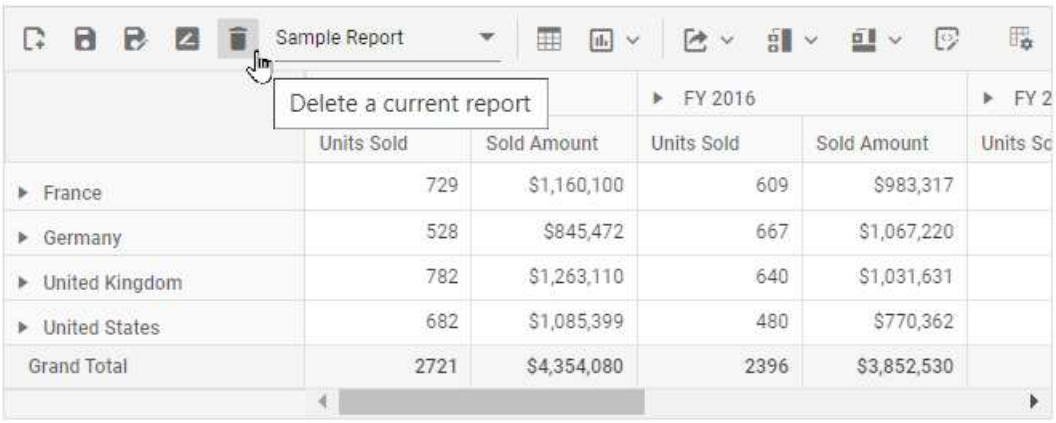
Copyright © 2001 -2024 Syncfusion Inc.

```
enableMultipleAxis: false,
},
toolbar: [
"New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid", "Chart", "MDX", "Export", "SubTotal", "GrandTotal",
"ConditionalFormatting", "FieldList"
],
allowExcelExport: true,
allowConditionalFormatting: true,
allowPdfExport: true,
showToolbar: true,
allowCalculatedField: true,
showFieldList: true,
width: "700",
};
},
methods: {
removeReport(args) {
fetch('https://localhost:44313/Pivot/RemoveReport', {
method: 'POST',
headers: {
'Accept': 'application/json',
'Content-Type': 'application/json',
},
body: JSON.stringify({ reportName: args.reportName })
}).then(response => {
this.fetchReport(args);
});
}
},
provide: {
pivotview: [FieldList, CalculatedField, Toolbar, ConditionalFormatting, NumberFormatting]
}
```

Copyright © 2001 -2024 Syncfusion Inc.

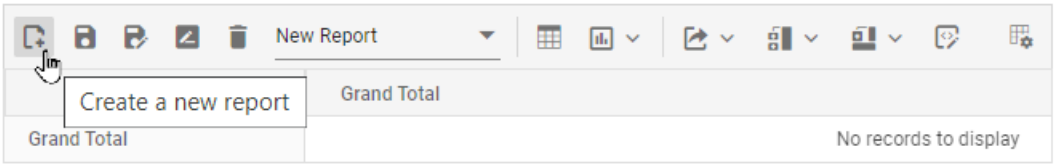
```
}
`
[PivotController.cs]
`csharp
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using System.Data;
namespace MyWebApp.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpPost]
[Route("Pivot/RemoveReport")]
public void RemoveReport([FromBody] Dictionary<string, string> reportArgs)
{
RemoveReportFromDB(reportArgs["reportName"]);
}
private void RemoveReportFromDB(string reportName)
{
SqlConnection sqlConn = OpenConnection();
SqlCommand cmd1 = null;
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
if ((row["ReportName"] as string).Equals(reportName))
{
cmd1 = new SqlCommand("delete from ReportTable where ReportName like '%" + reportName + "%'",
sqlConn);
break;
}
}
cmd1.ExecuteNonQuery();
```

```
sqlConn.Close();
}
private SqlConnection OpenConnection()
{
// Replace with your own connection string.
string connectionString = @"<Enter your valid connection string here>";
SqlConnection sqlConn = new SqlConnection(connectionString);
sqlConn.Open();
return sqlConn;
}
private DataTable GetDataTable(SqlConnection sqlConn)
{
string xquery = "select * from ReportTable";
SqlCommand cmd = new SqlCommand(xquery, sqlConn);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
return dt;
}
}
}
`
```



Adding a report

When you select the **"Create a new report"** option from the toolbar, the newReport event is triggered, followed by the saveReport event. To save this new report to the SQL database, use the saveReport event triggered later, and then follow the save report briefing in the preceding topic.



Limitations with respect to report manipulation

Below points need to be considered when saving the report to SQL Server database.

- **Data source**: Both raw data and aggregated data won't be saved and loaded from the database.
- **Hyperlinks**: Option to link external facts via pivot table cells won't be saved and loaded from the database.
- The pivot table should always load reports from the SQL database based on the data source that is currently bound to it.

In this GitHub repository, you can find our Vue Pivot Table sample and ASP.NET Core Web Application to save and load reports from SQL Server database.

Events

FetchReport

The event fetchReport is triggered when dropdown list is clicked in the toolbar in-order to retrieve and populate saved reports. It has following parameter - `reportName`. This event allows user to fetch the report names from local storage and populate the dropdown list.

LoadReport

The event loadReport is triggered when a report is selected from the dropdown list in the toolbar. It has following parameters - `report` and `reportName`. This event allows user to load the selected report to the pivot table.

NewReport

The event newReport is triggered when the new report icon is clicked in the toolbar. It has following parameter - `report`. This event allows user to create new report and add to the report list.

RenameReport

The event renameReport is triggered when rename report icon is clicked in the toolbar. It has following parameters - `rename`, `report` and `reportName`. This event allows user to rename the selected report from the report list.

RemoveReport

The event removeReport is triggered when remove report icon is clicked in the toolbar. It has following parameters - `report` and `reportName`. This event allows user to remove the selected report from the report list.

SaveReport

The event saveReport is triggered when save report icon is clicked in the toolbar. It has following parameters - `report` and `reportName`. This event allows user to save the altered report to the report list.

ToolbarRender

The toolbarRender event is triggered when the toolbar is rendered. It has the `customToolbar` parameter. This event helps to customize the built-in toolbar items and to include new toolbar item(s).

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:height="height" :showToolbar="showToolbar" :toolbar="toolbar"
:toolbarRender="beforeToolbarRender" :displayOption="displayOption"
:saveReport="saveReport" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import {
  PivotViewPlugin,
  IDataSet,
  Toolbar,
  FieldList
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: { columnWidth: 140 },
      displayOption: { view:'Both' },
      showToolbar: true,
      showFieldList: true,
      toolbar: [
        "Save",
        "Export",
        "FieldList"
      ]
    };
  },
  methods: {
     saveReport: function(args: any) {
      let reports = [];
      let isSaved = false;
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reports = JSON.parse(localStorage.pivotviewReports);
      }
      if (args.report && args.reportName && args.reportName !== "") {
```

```
        reports.map(function(item: any) {
          if (args.reportName === item.reportName) {
            item.report = args.report;
            isSaved = true;
          }
        });
        if (!isSaved) {
          reports.push(args);
        }
        localStorage.pivotviewReports = JSON.stringify(reports);
      }
    },
    beforeToolbarRender: function (args: any) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      args.customToolbar.splice(2, 0, {
          prefixIcon: 'e-rename-report e-icons', tooltipText: 'Custom
Button',
          click: this.customButton.bind(this),
        });
      args.customToolbar.splice(3, 0, {
            prefixIcon: 'e-pivotview-expand e-icons', tooltipText:
'Expand/Collapse',
            click: this.toolbarClicked.bind(this),
        });
        args.customToolbar[0].align = "Left";
        args.customToolbar[1].align = "Center";
        args.customToolbar[2].align = "Right";
    },
    customButton: function (args: any) {
      // Here you can customize the click event for custom button
    },
    toolbarClicked: function (args: any) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.dataSourceSettings.expandAll =
!pivotGridObj.dataSourceSettings.expandAll;
    }
  },
  provide: {
    pivotview: [
      Toolbar, FieldList
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs251" %}

Copyright © 2001 -2024 Syncfusion Inc.

BeforeExport

The pivot table (or) pivot chart can be exported as a pdf, excel, csv etc., document using the toolbar options. And, you can customize the export settings for exporting document by using the beforeExport event in the toolbar.

For example, you can add the header and footer for the pdf document by setting the header and footer properties for the pdfExportProperties in the beforeExport event.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:height="height" :allowExcelExport="allowExcelExport"
:allowConditionalFormatting="allowConditionalFormatting"
:allowPdfExport="allowPdfExport" :showToolbar="showToolbar"
:allowCalculatedField="allowCalculatedField" :showFieldList="showFieldList"
:toolbar="toolbar" :saveReport="saveReport" :loadReport="loadReport"
:fetchReport="fetchReport" :renameReport="renameReport"
:removeReport="removeReport" :newReport="newReport"
:beforeExport="beforeExport" :displayOption="displayOption"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import {
  PivotViewPlugin,
  GroupingBar,
  FieldList,
  IDataSet,
  CalculatedField,
  Toolbar,
  PDFExport,
  ExcelExport,
  ConditionalFormatting
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: { columnWidth: 140 },
      allowExcelExport: true,
```

```
      allowConditionalFormatting: true,
      allowPdfExport: true,
      displayOption: { view:'Both' },
      showToolbar: true,
      allowCalculatedField: true,
      showFieldList: true,
      toolbar: [
        "New",
        "Save",
        "SaveAs",
        "Rename",
        "Remove",
        "Load",
        "Grid",
        "Chart",
        "Export",
        "SubTotal",
        "GrandTotal",
        "ConditionalFormatting",
        "FieldList"
      ]
    };
  },
  methods: {
    saveReport: function(args: any) {
      let reports = [];
      let isSaved = false;
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reports = JSON.parse(localStorage.pivotviewReports);
      }
      if (args.report && args.reportName && args.reportName !== "") {
        reports.map(function(item: any) {
          if (args.reportName === item.reportName) {
            item.report = args.report;
            isSaved = true;
          }
        });
        if (!isSaved) {
          reports.push(args);
        }
        localStorage.pivotviewReports = JSON.stringify(reports);
      }
    },
    fetchReport: function(args: any) {
      let reportCollection = [];
      let reportList: any = [];
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
      }
      reportCollection.map(function(item: any) {
        reportList.push(item.reportName);
```

```
      });
      args.reportName = reportList;
    },
    loadReport: function(args: any) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let reportCollection = [];
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
      }
      reportCollection.map(function(item: any) {
        if (args.reportName === item.reportName) {
          args.report = item.report;
        }
      });
      if (args.report) {
        pivotGridObj.dataSourceSettings =
JSON.parse(args.report).dataSourceSettings;
      }
    },
    removeReport: function(args: any) {
      let reportCollection = [];
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
      }
      for (let i = 0; i < reportCollection.length; i++) {
        if (reportCollection[i].reportName === args.reportName) {
          reportCollection.splice(i, 1);
        }
      }
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        localStorage.pivotviewReports = JSON.stringify(reportCollection);
      }
    },
    renameReport: function(args: any) {
      let reportCollection = [];
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
      }
      reportCollection.map(function(item: any) {
        if (args.reportName === item.reportName) {
          item.reportName = args.rename;
        }
      });
      if (
```

```
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        localStorage.pivotviewReports = JSON.stringify(reportCollection);
      }
    },
    newReport: function() {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.setProperties(
        {
          dataSourceSettings: {
            columns: [],
            rows: [],
            values: [],
            filters: []
          }
        },
        false
      );
    },
    beforeExport: function(args: any): void {
            args.excelExportProperties = {
                header: {
                    headerRows: 2,
                    rows: [
                        { cells: [{ colSpan: 4, value: "Pivot Table", style:
{ fontColor: '#C67878', fontSize: 20, hAlign: 'Center', bold: true,
underline: true } }] }
                    ]
                },
                footer: {
                    footerRows: 4,
                    rows: [
                        { cells: [{ colSpan: 4, value: "Thank you for your
business!", style: { hAlign: 'Center', bold: true } }] },
                        { cells: [{ colSpan: 4, value: "!Visit Again!",
style: { hAlign: 'Center', bold: true } }] }
                    ]
                }
            };
            args.pdfExportProperties = {
                header: {
                    fromTop: 0,
                    height: 130,
                    contents: [
                        {
                            type: 'Text',
                            value: "Pivot Table",
                            position: { x: 0, y: 50 },
                            style: { textBrushColor: '#000000', fontSize:
13, dashStyle:'Solid',hAlign:'Center' }
                        }
                    ]
                },
                footer: {
                    fromBottom: 160,
```

```
                    height: 150,
                    contents: [
                        {
                            type: 'PageNumber',
                            pageNumberType: 'Arabic',
                            format: 'Page {$current} of {$total}',
                            position: { x: 0, y: 25 },
                            style: { textBrushColor: '#02007a', fontSize: 15
}
                        }
                    ]
                }
            };
        }
  },
  provide: {
    pivotview: [
      FieldList,
      CalculatedField,
      Toolbar,
      PDFExport,
      ExcelExport,
      ConditionalFormatting
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs252" %}

ActionBegin

The event actionBegin triggers when the UI actions such as switching between pivot table and pivot chart, changing chart types, conditional formatting, exporting, etc. that are present in toolbar UI begin. This allows user to identify the current action being performed at runtime. It has the following parameters:

- dataSourceSettings: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- actionName: It holds the name of the current action began. The following are the UI actions and their names:

| Action | Action Name |

|------|------------|

| New report | Add new report |

| Save report | Save current report |

| Save as report | Save as current report |

| Rename report | Rename current report |

Copyright © 2001 -2024 Syncfusion Inc.

| Remove report | Remove current report |

| Report change | Report change |

| Conditional Formatting | Open conditional formatting dialog |

| Number Formatting | Open number formatting dialog |

| Export menu | PDF export, Excel export, CSV export |

| Show Fieldlist | Open field list |

| Show Table | Show table view |

| Chart menu | Show chart view |

| Sub-totals menu | Hide sub-totals, Show row sub-totals, Show column sub-totals, Show sub-totals |

| Grand totals menu | Hide grand totals, Show row grand totals, Show column grand totals, Show grand totals |

- cancel: It allows user to restrict the current action.

In the below sample, toolbar UI actions such as add new report and save current report can be restricted by setting the **args.cancel** option to **true** in the actionBegin event.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:height="height"  :allowExcelExport="allowExcelExport"
:allowConditionalFormatting="allowConditionalFormatting"
:allowPdfExport="allowPdfExport" :showToolbar="showToolbar"
:allowNumberFormatting="allowNumberFormatting"
:allowCalculatedField="allowCalculatedField" :showFieldList="showFieldList"
:toolbar="toolbar" :actionBegin="actionBegin"
:displayOption="displayOption"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import {
  PivotViewPlugin,
  GroupingBar,
  FieldList,
  IDataSet,
  CalculatedField,
  Toolbar,
  PDFExport,
  ExcelExport,
  ConditionalFormatting,
  NumberFormatting,
  PivotActionBeginEventArgs
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
```

```
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: { columnWidth: 140 },
      allowExcelExport: true,
      allowConditionalFormatting: true,
      allowNumberFormatting: true,
      allowPdfExport: true,
      displayOption: { view:'Both' },
      showToolbar: true,
      allowCalculatedField: true,
      showFieldList: true,
      toolbar: [
        "New",
        "Save",
        "SaveAs",
        "Rename",
        "Remove",
        "Load",
        "Grid",
        "Chart",
        "Export",
        "SubTotal",
        "GrandTotal",
        "ConditionalFormatting",
        "NumberFormatting",
        "FieldList"
      ]
    };
  },
  methods: {
    actionBegin: function (args: PivotActionBeginEventArgs) {
      if (args.actionName == 'Add new report' || args.actionName == 'Save
current report') {
        args.cancel = true;
      }
    }
  },
  provide: {
    pivotview: [
      FieldList,
      CalculatedField,
      Toolbar,
      PDFExport,
      ExcelExport,
```

```
      ConditionalFormatting,
      NumberFormatting
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs253" %}

ActionComplete

The event actionComplete triggers when the UI actions such as as switching between pivot table and pivot chart, changing chart types, conditional formatting, exporting, etc. that are present in toolbar UI, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- dataSourceSettings: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- actionName: It holds the name of the current action completed. The following are the UI actions and their names:

Action	Action Name
New report	New report added
Save report	Report saved
Save as report	Report re-saved
Rename report	Report renamed
Remove report	Report removed
Report change	Report changed
Conditional Formatting	Conditionally formatted
Number Formatting	Number formatted
Export menu	PDF exported, Excel exported, CSV exported
Show Fieldlist	Field list closed
Show Table	Table view shown
Sub-totals menu	Sub-totals hidden, Row sub-totals shown, Column sub-totals shown, Sub-totals shown
Grand totals menu	Grant totals hidden, Row grand totals shown, Column grand totals shown, Grand totals shown

Copyright © 2001 -2024 Syncfusion Inc.

- actionInfo: It holds the unique information about the current UI action. For example, while adding new report, the event argument contains information such as report name and the action name.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:height="height"  :allowExcelExport="allowExcelExport"
:allowConditionalFormatting="allowConditionalFormatting"
:allowPdfExport="allowPdfExport" :showToolbar="showToolbar"
:allowNumberFormatting="allowNumberFormatting"
:allowCalculatedField="allowCalculatedField" :showFieldList="showFieldList"
:toolbar="toolbar" :actionComplete="actionComplete"
:displayOption="displayOption"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import {
  PivotViewPlugin,
  GroupingBar,
  FieldList,
  IDataSet,
  CalculatedField,
  Toolbar,
  PDFExport,
  ExcelExport,
  ConditionalFormatting,
  NumberFormatting,
  PivotActionCompleteEventArgs
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: { columnWidth: 140 },
      allowExcelExport: true,
      allowConditionalFormatting: true,
      allowNumberFormatting: true,
      allowPdfExport: true,
```

```
      displayOption: { view:'Both' },
      showToolbar: true,
      allowCalculatedField: true,
      showFieldList: true,
      toolbar: [
        "New",
        "Save",
        "SaveAs",
        "Rename",
        "Remove",
        "Load",
        "Grid",
        "Chart",
        "Export",
        "SubTotal",
        "GrandTotal",
        "ConditionalFormatting",
        "NumberFormatting",
        "FieldList"
      ]
    };
  },
  methods: {
    actionComplete: function (args: PivotActionCompleteEventArgs) {
      if (args.actionName == 'New report added' || args.actionName ==
'Report saved') {
        // Triggers when the toolbar UI actions such as add new report and
save current report icon are completed.
      }
    }
  },
  provide: {
    pivotview: [
      FieldList,
      CalculatedField,
      Toolbar,
      PDFExport,
      ExcelExport,
      ConditionalFormatting,
      NumberFormatting
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs254" %}

ActionFailure

The event actionFailure triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- actionName: It holds the name of the current action failed. The following are the UI actions and their names:

| Action | Action Name |

|------|------------|

| New report | Add new report |

| Save report | Save current report |

| Save as report | Save as current report |

| Rename report | Rename current report |

| Remove report | Remove current report |

| Report change | Report change |

| Conditional Formatting | Open conditional formatting dialog |

| Number Formatting | Open number formatting dialog |

| Export menu | PDF export, Excel export, CSV export |

| Show Fieldlist | Open field list |

| Show Table | Show table view |

| Chart menu | Show chart view |

| Sub-totals menu | Hide sub-totals, Show row sub-totals, Show column sub-totals, Show sub-totals |

| Grand totals menu | Hide grand totals, Show row grand totals, Show column grand totals, Show grand totals |

- errorInfo: It holds the error information of the current UI action.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:height="height"  :allowExcelExport="allowExcelExport"
:allowConditionalFormatting="allowConditionalFormatting"
:allowPdfExport="allowPdfExport" :showToolbar="showToolbar"
:allowNumberFormatting="allowNumberFormatting"
:allowCalculatedField="allowCalculatedField" :showFieldList="showFieldList"
:toolbar="toolbar" :actionFailure="actionFailure"
:displayOption="displayOption"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import {
  PivotViewPlugin,
  GroupingBar,
  FieldList,
```

```
  IDataSet,
  CalculatedField,
  Toolbar,
  PDFExport,
  ExcelExport,
  ConditionalFormatting,
  NumberFormatting,
  PivotActionFailureEventArgs
} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: { columnWidth: 140 },
      allowExcelExport: true,
      allowConditionalFormatting: true,
      allowNumberFormatting: true,
      allowPdfExport: true,
      displayOption: { view:'Both' },
      showToolbar: true,
      allowCalculatedField: true,
      showFieldList: true,
      toolbar: [
        "New",
        "Save",
        "SaveAs",
        "Rename",
        "Remove",
        "Load",
        "Grid",
        "Chart",
        "Export",
        "SubTotal",
        "GrandTotal",
        "ConditionalFormatting",
        "NumberFormatting",
        "FieldList"
      ]
    };
  },
  methods: {
    actionFailure: function (args: PivotActionFailureEventArgs) {
      if (args.actionName == 'Add new report' || args.actionName == 'Save
current report') {
```

```
            // Triggers when the current UI action fails to achieve the
desired result.
        }
    }
  },
  provide: {
    pivotview: [
      FieldList,
      CalculatedField,
      Toolbar,
      PDFExport,
      ExcelExport,
      ConditionalFormatting,
      NumberFormatting
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs255" %}

See Also

- Toolbar Component

Tool tip in Vue Pivotview component

The tooltip can be enabled or disabled by setting the showTooltip property to **true** or **false**. By default, tooltip is enabled in the pivot table.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :height="height"
:dataSourceSettings="dataSourceSettings" :showTooltip="showTooltip"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
```

```
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showTooltip: false
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs256" %}

Tooltip Template

User can design their own tooltip by setting the property tooltipTemplate with own HTML elements. The property accepts both HTML string and ID attribute. The following place holders are available to display its dynamic values inside the HTML elements.

${rowHeaders} – Row headers of the selected value cell.

${columnHeaders} – Column headers of the selected value cell.

${rowFields} – Row fields of the selected value cell.

${columnFields} – Column fields of the selected value cell.

${valueField} – Field name of the selected value cell.

${aggregateType} – Aggregate type of the selected value cell.

${value} - Formatted value of the selected value cell.

The tooltip customization is common for both pivot table and pivot chart or it can be done individually as well. To customize the pivot table tooltip, the above procedure needs to be followed. To customize the pivot chart tooltip alone use template property of tooltip under chartSettings.

In the below sample, the pivot table and pivot chart shows customized tooltip layouts.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:displayOption="displayOption" :chartSettings="chartSettings"
:toolbar="toolbar" :showToolbar="showToolbar"
:tooltipTemplate="tooltipTemplate"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { PivotViewPlugin, Toolbar} from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: []
    },
    height: 350,
    displayOption: { view: 'Both' },
    chartSettings: {
        value: 'Amount', enableExport: true, chartSeries: { type: 'Column',
animation: { enable: false } }, enableMultipleAxis: false,
        tooltip: { template: '<span
class="pivotTooltipTemplateWrap">${aggregateType} of ${valueField}:
${value}</span>' }
    },
        toolbar: ['Grid', 'Chart'],
        showToolbar: true,
        tooltipTemplate: `<div class='pivotTooltipTemplateWrap'><span
class='pivotTooltipHeader'>`+"${columnHeaders}"+`:</span ><span
class='pivotTooltipValue'>`+"${value}"+`</span></div>`
    }
  },
  provide: {
      pivotview: [Toolbar]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
.pivotTooltipTemplateWrap {
  border: 3px solid #27b1f0;
  background-color: #4d4d4d;
  width: auto;
  color: #FFFFFF;
  padding: 5px;
  font-size: 12px;
}
.pivotTooltipValue {
  font-style: italic;
}
.pivotTooltipHeader {
  color: aqua;
  font-weight: bold;
  width: 100px;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs257" %}

Css customization in Vue Pivotview component

Hiding Axis

The visibility of row, column, value and filter axis in Field List and Grouping Bar can be changed using custom CSS setting. To do so, please refer the code sample below:

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="PivotTable" :height="height"
:dataSourceSettings="dataSourceSettings" :showGroupingBar="showGroupingBar"
:showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, GroupingBar, FieldList } from "@syncfusion/ej2-
vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showGroupingBar: true,
      showFieldList: true
    }
  },
  provide: {
    pivotview: [GroupingBar,FieldList]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
/* csslint ignore:start */
#PivotTable .e-group-columns {
  display: none;
}
#PivotTable .e-group-filters {
```

```
  height: 71px !important;
}
#PivotTable_PivotFieldList_Wrapper .e-field-list-columns{
  display: none;
}
#PivotTable_PivotFieldList_Wrapper .e-field-list-values{
  margin-top: 0px;
  height: 338px;
}
.e-pivotfieldlist-wrapper .e-values {
  height: 310px !important;
}
/* Hiding row axis in grouping bar */
/* #PivotView .e-group-rows {
    display: none;
} */
/* Hiding row axis in field list */
/* .e-pivotfieldlist-wrapper .e-field-list-rows {
    display: none;
} */
/* Hiding value axis in grouping bar */
/* #PivotView .e-group-values {
    display: none;
} */
/* Hiding value axis in field list */
/* .e-pivotfieldlist-wrapper .e-field-list-values {
    display: none;
} */
/* Hiding filter axis in grouping bar */
/* #PivotView .e-group-filters {
    display: none;
} */
/* Hiding filter axis in field list */
/* .e-pivotfieldlist-wrapper .e-field-list-filters {
    display: none;
} */
/* csslint ignore:end */
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs16" %}

Text Alignment

The alignment of text inside row headers, column headers, value cells and summary cells can be changed using custom CSS setting. To do so, please refer the code sample below:

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :showGroupingBar="showGroupingBar"
:showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { PivotViewPlugin, GroupingBar, FieldList } from "@syncfusion/ej2-
vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        enableSorting: true,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showGroupingBar: true,
      showFieldList: true
    }
  },
  provide: {
    pivotview: [GroupingBar,FieldList]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
/* csslint ignore:start */
.e-pivotview .e-valuescontent {
  text-align: center !important;
}
/* csslint ignore:end */
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs17" %}

Customize header, value and summary cell style

The elements in pivot table like header cell, value cell and summary cell style can be customized using built-in CSS names. To do so, please refer the code sample below:

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="PivotTable" :height="height"
:dataSourceSettings="dataSourceSettings" :showGroupingBar="showGroupingBar"
:showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
```

```
import Vue from "vue";
import { PivotViewPlugin, GroupingBar, FieldList } from "@syncfusion/ej2-
vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        drilledMembers: [{ name: 'Year', items: ['FY 2015'] }, { name:
'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showGroupingBar: true,
      showFieldList: true
    }
  },
  provide: {
    pivotview: [GroupingBar,FieldList]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
/* csslint ignore:start */
.e-pivotview .e-headercell {
  background-color: thistle !important;
}
.e-pivotview .e-rowsheader {
  background-color: skyblue !important;
}
.e-pivotview .e-valuescontent:not(.e-gtot) {
  background-color: pink !important;
}
.e-pivotview .e-gtot {
  background-color: greenYellow !important;
}
/* csslint ignore:end */
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs18" %}

Printing and Exporting

Print in Vue Pivotview component

The rendered pivot table can be printed directly from the browser by invoking the print method from the grid's instance. The below sample code illustrates the print option being invoked by an external button click.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="print-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick" >Print</ejs-button>
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height" >
        </ejs-pivotview>
</div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        rows: [{ name: 'Country' }, { name: 'Products' }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Amount', caption: 'Sold Amount' }, { name: 'Sold',
caption: 'Units Sold' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
      },
      height: 350,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.grid.print();
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs270" %}

Similarly, to print the pivot chart, use the print method from the chart's instance. The below sample code illustrates the print option being invoked by an external button click.

To use pivot chart, you need to inject the PivotChart module in the pivot table.

To display the pivot chart, set the displayOption property to either **Chart** or **Both**.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="print-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick" >Print</ejs-button>
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:displayOption="displayOption" >
        </ejs-pivotview>
</div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        rows: [{ name: 'Country' }, { name: 'Products' }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Amount', caption: 'Sold Amount' }, { name: 'Sold',
caption: 'Units Sold' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
      },
      height: 350,
      displayOption: { view: 'Chart' },
      isPrimary: true,
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.chart.print();
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs271" %}

Copyright © 2001 -2024 Syncfusion Inc.

Excel export in Vue Pivotview component

The Excel export allows pivot table data to export as Excel document. To enable Excel export in the pivot table, set the allowExcelExport as **true**. You need to use the excelExport method for Excel exporting.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Excel Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowExcelExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.excelExport();
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs48" %}

Copyright © 2001 -2024 Syncfusion Inc.

Multiple pivot table exporting

The Excel export provides an option to export multiple pivot table data in the same Excel file.

Same WorkSheet

The Excel export provides support to export multiple pivot tables in same sheet. To export in same sheet, define multipleExport.type as AppendToSheet in ExcelExportProperties. It has an option to provide blank rows between pivot tables and these blank row(s) count can be defined using themultipleExport.blankRows property.

By default, multipleExport.blankRows value is 5 between pivot tables within the same sheet.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Excel Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
        <ejs-pivotview id="pivotview2" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowExcelExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
```

```
      let pivotGridObj2 =
document.getElementById('pivotview2').ej2_instances[0];
      let excelExportProperties: ExcelExportProperties = {
          multipleExport: { type: 'AppendToSheet', blankRows: 2 }
      };
      let firstGridExport: Promise<any> =
pivotGridObj.grid.excelExport(excelExportProperties, true);
      firstGridExport.then((fData: any) => {
          pivotGridObj2.excelExport(excelExportProperties, false, fData);
      });
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
#pivotview2 {
  margin-top: 20px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs49" %}

New WorkSheet

Excel export provides support to export multiple pivot tables into new sheets. To export in new sheets, define `multipleExport.type` as `NewSheet` in `ExcelExportProperties`.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Excel Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
        <ejs-pivotview id="pivotview2" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
```

```
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowExcelExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let pivotGridObj2 =
document.getElementById('pivotview2').ej2_instances[0];
      let excelExportProperties: ExcelExportProperties = {
          multipleExport: { type: 'NewSheet' }
      };
      let firstGridExport: Promise<any> =
pivotGridObj.grid.excelExport(excelExportProperties, true);
      firstGridExport.then((fData: any) => {
          pivotGridObj2.excelExport(excelExportProperties, false, fData);
      });
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
#pivotview2 {
  margin-top: 20px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs50" %}

Changing the pivot table style while exporting

The Excel export provides an option to change colors for headers, caption and records in pivot table before exporting. In-order to apply colors, define **theme** settings in **excelExportProperties** object and pass it as a parameter to the excelExport method.

By default, material theme is applied to exported Excel document.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Excel Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
```

```
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowExcelExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let excelExportProperties: ExcelExportProperties = {
        theme:
          {
              header: { fontName: 'Segoe UI', fontColor: '#666666' },
              record: { fontName: 'Segoe UI', fontColor: '#666666' },
              caption: { fontName: 'Segoe UI', fontColor: '#666666' }
          }
        };
      pivotGridObj.excelExport(excelExportProperties);
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs51" %}

Copyright © 2001 -2024 Syncfusion Inc.

Add header and footer while exporting

The Excel export provides an option to include header and footer content for the excel document before exporting. In-order to add header and footer, define **header** and **footer** properties in **excelExportProperties** object and pass it as a parameter to the excelExport method.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Excel Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowExcelExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let excelExportProperties: ExcelExportProperties = {
          header: {
              headerRows: 2,
              rows: [
                  { cells: [{ colSpan: 4, value: "Pivot Table", style: {
fontColor: '#C67878', fontSize: 20, hAlign: 'Center', bold: true, underline:
true } }] }
              ]
          },
```

```
            footer: {
                footerRows: 4,
                rows: [
                    { cells: [{ colSpan: 4, value: "Thank you for your
business!", style: { hAlign: 'Center', bold: true } }] },
                    { cells: [{ colSpan: 4, value: "!Visit Again!", style: {
hAlign: 'Center', bold: true } }] }
                ]
            }
        };
        pivotGridObj.excelExport(excelExportProperties);
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs52" %}

Changing the file name while exporting

The Excel export provides an option to change file name of the document before exporting. In-order to change the file name, define **fileName** property in **excelExportProperties** object and pass it as a parameter to the excelExport method.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Excel Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
```

```
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowExcelExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let excelExportProperties: ExcelExportProperties = {
        fileName: 'sample.xlsx'
        };
      pivotGridObj.excelExport(excelExportProperties);
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs53" %}

CSV Export

Also, the Excel export allows pivot table data to be exported in CSV file format. To export pivot table in CSV file format, you need to use the csvExport method.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Excel Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
```

```
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowExcelExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.csvExport();
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs54" %}

Virtual Scroll Data

You can export the pivot table virtual scroll data as Excel/CSV document by using PivotEngine export without any performance degradation. To enable PivotEngine export in the pivot table, set the allowExcelExport as **true**. You need to use the exportToExcel method for PivotEngine export.

To use PivotEngine export, You need to inject the ExcelExport module in pivot table.

PivotEngine export will be performed while enabling virtual scrolling by default.

Virtual Scroll Data Excel Export

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Excel Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, ExcelExport } from "@syncfusion/ej2-vue-
pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowExcelExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.excelExportModule.exportToExcel('Excel');
    }
  },
  provide: {
    pivotview: [
      ExcelExport
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs55" %}

Virtual Scroll Data CSV Export

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Excel Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { PivotViewPlugin, ExcelExport } from "@syncfusion/ej2-vue-
pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowExcelExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
       let excelExportProperties = {
        fileName: 'csvexport.csv',
      };
      pivotGridObj.csvExport(excelExportProperties);
    }
  },
  provide: {
    pivotview: [
      ExcelExport
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs56" %}

Export all pages

The pivot engine exports the entire virtual data of the pivot table (i.e. the data that contains all of the records used to render the complete pivot table) as an Excel/CSV document. To export just the current viewport of the pivot table, set the exportAllPages property to **false**. To use the pivot engine export, add the ExcelExport module into the pivot table.

By default, the pivot engine export will be performed while virtual scrolling is enabled.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Excel Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"
:enableVirtualization="enableVirtualization"
exportAllPages="exportAllPages"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, VirtualScroll, } from "@syncfusion/ej2-vue-
pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        rows: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        values: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowExcelExport: true,
      enableVirtualization: true,
      exportAllPages: true,
      isPrimary: true
    }
  },
  provide: {
    pivotview: [VirtualScroll]
  }
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.excelExport();
    }
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs57" %}

Events

ExcelQueryCellInfo

The event excelQueryCellInfo triggers while framing each row and value cell during Excel export. It allows the user to customize the cell value, style etc. of the current cell. It has the following parameters:

- value - It holds the cell value.
- column - It holds column information for the current cell.
- data - It holds the entire row data across the current cell.
- style - It holds the style properties for the cell.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      allowExcelExport: true,
      gridSettings: {
        columnWidth: 140,
        excelQueryCellInfo: function(args){
            //triggers for every header cell while Excel/CSV export
        }
      }
    }
```

```
  }
  methods:{
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.excelExport();
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs58" %}

ExcelHeaderQueryCellInfo

The event excelHeaderQueryCellInfo triggers on framing each header cell during Excel export. It allows the user to customize the cell value, style etc. of the current cell. It has the following parameters:

- cell - It holds the current cell information.
- style - It holds the style properties for the cell.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"
:allowExcelExport="allowExcelExport"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      allowExcelExport: true,
      gridSettings: {
```

```
        columnWidth: 140,
        excelHeaderQueryCellInfo: function(args){
            //triggers for every header cell while Excel/CSV export
        }
      }
    }
  }
  methods:{
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.excelExport();
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs59" %}

ExportComplete

The event exportComplete is triggered after the pivot table data has been exported to an Excel/CSV document. You can use this event to acquire blob stream data for further customization and processing at your end by passing the isBlob parameter as **true** when using the excelExport method. It has the following parameters:

- type - It holds the current export type such as PDF, Excel, and CSV.
- promise - It holds the promise object for blob data.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Excel Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings"
:allowExcelExport="allowExcelExport"
:enableVirtualization="enableVirtualization"
:exportComplete="exportComplete"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, VirtualScroll, ExportCompleteEventArgs,
ExcelExport } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        rows: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        values: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowExcelExport: true,
      enableVirtualization: true,
      isPrimary: true
    }
  },
  provide: {
    pivotview: [
      VirtualScroll,
      ExcelExport
    ]
  }
  methods: {
    exportComplete: function (args: ExportCompleteEventArgs) {
      args.promise.then( function(e: { blobData: Blob }) {
        console.log(e.blobData);
      });
    },
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let excelExportProperties: ExcelExportProperties = {
        fileName: 'excelexport.xlsx'
      };
      pivotGridObj.excelExport(excelExportProperties, false, null, true);
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs60" %}

See Also

- PDF Exporting

Copyright © 2001 -2024 Syncfusion Inc.

Pdf export in Vue Pivotview component

PDF export allows exporting pivot table data as PDF document. To enable PDF export in the pivot table, set the allowPdfExport as **true**. You need to use the pdfExport method for PDF exporting.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.pdfExport();
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs160" %}

Copyright © 2001 -2024 Syncfusion Inc.

Multiple pivot table exporting

PDF export provides an option for exporting multiple pivot tables to same file. In this exported document, each pivot table will be exported to new page of document in same file.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport">
</ejs-pivotview>
        <ejs-pivotview id="pivotview2" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let pivotGridObj2 =
document.getElementById('pivotview2').ej2_instances[0];
      let firstGridPdfExport: Promise<Object> =
pivotGridObj.grid.pdfExport({}, true);
      firstGridPdfExport.then((pdfData: Object) => {
          pivotGridObj2.pdfExport({}, false, pdfData);
      });
    }
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
#pivotview2 {
  margin-top: 20px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs161" %}

Export table and chart into the same document

When the displayOption is set to **Both**, you can export both the table and the chart into the same PDF document. To achieve this, use the pdfExport method and set the exportBothTableAndChart parameter to **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport"
        :displayOption="displayOption" :chartSettings="chartSettings">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PDFExport } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      isPrimary: true,
      displayOption: { view: 'Both', primary: 'Table' },
      chartSettings: {
        value: 'Amount', enableExport: true, chartSeries: { type: 'Column' }
```

```
      }
    }
  },
  provide: {
    pivotview: [
      PDFExport
    ]
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.pdfExport(null, false, null, false, true);
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs162" %}

Customization during PDF export

PDF export provides option to customize mapping of pivot table to the exported PDF document.

To add header and footer while exporting

You can customize text, page number, line, page size and changing orientation in header and footer of the exported document.

To add a text in header/footer

You can add text either in header or footer of the exported PDF document like in the below code example.

`ts

let pdfExportProperties: PdfExportProperties = {

header: {

fromTop: 0,

height: 130,

contents: [

{

type: 'Text',

value: "Northwind Traders",

position: { x: 0, y: 50 },

style: { textBrushColor: '#000000', fontSize: 13 }

},

]

```
}
}
```

To draw a line in header/footer

You can add line either in header or footer of the exported PDF document like in the below code example.

Supported line styles:

- dash
- dot
- dashdot
- dashdotdot
- solid

```ts
let pdfExportProperties: PdfExportProperties = {
header: {
fromTop: 0,
height: 130,
contents: [
{
type: 'Line',
style: { penColor: '#000080', penSize: 2, dashStyle: 'Solid' },
points: { x1: 0, y1: 4, x2: 685, y2: 4 }
}
]
}
}
```

Add page number in header/footer

You can add page number either in header or footer of exported PDF document like in the below code example.

Supported page number types:

- LowerLatin - a, b, c,
- UpperLatin - A, B, C,
- LowerRoman - i, ii, iii,
- UpperRoman - I, II, III,
- Number - 1,2,3.

Copyright © 2001 -2024 Syncfusion Inc.

`ts

let pdfExportProperties: PdfExportProperties = {

header: {

fromTop: 0,

height: 130,

contents: [

{

type: 'PageNumber',

pageNumberType: 'Arabic',

format: 'Page {$current} of {$total}', //optional

position: { x: 0, y: 25 },

style: { textBrushColor: '#ffff80', fontSize: 15, hAlign: 'Center' }

}

]

}

}

`

The below code illustrates the PDF export customization options.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
```

```
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let pdfExportProperties: PdfExportProperties = {
        header: {
            fromTop: 0,
            height: 130,
            contents: [
                {
                    type: 'Text',
                    value: "Pivot Table",
                    position: { x: 0, y: 50 },
                    style: { textBrushColor: '#000000', fontSize: 13,
dashStyle:'Solid',hAlign:'Center' }
                }
            ]
        },
        footer: {
            fromBottom: 160,
            height: 150,
            contents: [
                {
                    type: 'PageNumber',
                    pageNumberType: 'Arabic',
                    format: 'Page {$current} of {$total}',
                    position: { x: 0, y: 25 },
                    style: { textBrushColor: '#02007a', fontSize: 15 }
                }
            ]
        }
    };
    pivotGridObj.pdfExport(pdfExportProperties);
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs163" %}

Add an image in header/footer

You can add image (Base64 string) either in header or footer of the exported PDF document like in the below code example.

`ts

let pdfExportProperties: PdfExportProperties = {

header: {

fromTop: 0,

height: 130,

contents: [

{

type: 'Image',

src: image,

position: { x: 20, y: 10 },

size: { height: 100, width: 100 },

}

]

}

}

`

The below code illustrates the PDF export customization options.

APP.VUE

```
<template>
  <div id="app">
    <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
    <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport">
</ejs-pivotview>
  </div>
</template>
<script>
  import Vue from "vue";
  import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
  import { PdfExportProperties } from '@syncfusion/ej2-grids';
  import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-
buttons";
  import { pivotData } from './pivotData.js';
  Vue.use(PivotViewPlugin);
  Vue.use(ButtonPlugin);
  export default {
    data () {
      return {
        dataSourceSettings: {
```

```
            dataSource: pivotData,
            expandAll: false,
            drilledMembers: [{ name: 'Country', items: ['France'] }],
            columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
            values: [{ name: 'Sold', caption: 'Units Sold' }, { name:
'Amount', caption: 'Sold Amount' }],
            rows: [{ name: 'Country' }, { name: 'Products' }],
            formatSettings: [{ name: 'Amount', format: 'C0' }],
            filters: []
          },
          height: 320,
          allowPdfExport: true,
          isPrimary: true
        }
      },
      methods: {
        btnClick: function(args) {
          let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
          let pdfExportProperties: PdfExportProperties = {
            header: {
              fromTop: 0,
              height: 130,
              contents: [
                {
                  type: 'Image',
                  src: image,
                  position: { x: 20, y: 10 },
                  size: { height: 100, width: 100 },
                }
              ]
            }
          };
          pivotGridObj.pdfExport(pdfExportProperties);
        }
      }
    }
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs274" %}

Changing the file name while exporting

The PDF export provides an option to change file name of the document before exporting. In-order to change the file name, define **fileName** property in **pdfExportProperties** object and pass it as a parameter to the pdfExport method.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
```

```
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let pdfExportProperties: PdfExportProperties = {
        filename:'sample.pdf'
      };
      pivotGridObj.pdfExport(pdfExportProperties);
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs164" %}

Changing page orientation while exporting

The PDF export provides an option to change page orientation of the document before exporting. In-order to change the page orientation, define **pageOrientation** property in **pdfExportProperties** object

Copyright © 2001 -2024 Syncfusion Inc.

and pass it as a parameter to the pdfExport method. By default, the page orientation will be in **Portrait** and it can be changed to **Landscape** based on user requirement.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      allowPdfExport: true,
      height: 320,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let pdfExportProperties: PdfExportProperties = {
        pageOrientation: 'Landscape'
      };
      pivotGridObj.pdfExport(pdfExportProperties);
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs165" %}

Changing page size while exporting

The PDF export provides an option to change page size of the document before exporting. In-order to change the page size, define **pageSize** property in **pdfExportProperties** object and pass it as a parameter to the pdfExport method.

Supported page sizes are: Letter, Note, Legal, A0, A1, A2, A3, A5, A6, A7, A8, A9, B0, B1, B2, B3, B4, B5, Archa, Archb, Archc, Archd, Arche, Flsa, HalfLetter, Letter11x17, Ledger.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let pdfExportProperties: PdfExportProperties = {
          pageSize: 'Letter'
```

```
        };
        pivotGridObj.pdfExport(pdfExportProperties);
      }
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs166" %}

Changing document width and height while exporting

Before exporting, you can change the height and width of the PDF document. To achieve this, use the **height** and **width** properties in the beforeExport event.

This option is only available if enableVirtualization is set to **true**. In addition, the VirtualScroll and PDFExport modules must be injected into the pivot table.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport"
:enableVirtualization="enableVirtualization" :beforeExport="beforeExport">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PDFExport, BeforeExportEventArgs, VirtualScroll }
from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
```

```
      enableVirtualization: true,
      isPrimary: true
    }
  },
  provide: {
    pivotview: [
      PDFExport,
      VirtualScroll
    ]
  },
  methods: {
    beforeExport: function (args: BeforeExportEventArgs) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      args.width = pivotGridObj.element.offsetWidth;
      args.height = pivotGridObj.element.offsetHeight;
    },
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.pdfExport();
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs167" %}

Customize the table column count while exporting

Before exporting, you can split and export the pivot table columns on each page of the PDF document by using the **columnSize** property in the beforeExport event.

This option is only available if enableVirtualization is set to **true**. In addition, the VirtualScroll and PDFExport modules must be injected into the pivot table.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport"
:enableVirtualization="enableVirtualization" :beforeExport="beforeExport">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PDFExport, BeforeExportEventArgs, VirtualScroll }
from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
```

```
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      enableVirtualization: true,
      isPrimary: true
    }
  },
  provide: {
    pivotview: [
      PDFExport,
      VirtualScroll
    ]
  },
  methods: {
    beforeExport: function (args: BeforeExportEventArgs) {
      args.columnSize = 6;
    },
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.pdfExport();
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs168" %}

Changing the table's column width and row height while exporting

You can change the column width and row height in the PDF document during the pivot table export by using the onPdfCellRender event. Within this event, the args.column.width property allows you to change the width of specific columns.

As shown in the code example below, the **“Unit Sold”** column under **“FY 2015”** is changed to a width of **60** pixels.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport"
        :enableVirtualization="enableVirtualization"
:onPdfCellRender="onPdfCellRender"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PDFExport, PdfCellRenderArgs, VirtualScroll } from
"@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      isPrimary: true,
      enableVirtualization: true,
    }
  },
  methods: {
    onPdfCellRender: function (args: PdfCellRenderArgs) {
      if (args.pivotCell && args.pivotCell.valueSort &&
args.pivotCell.valueSort.levelName === 'FY 2015.Units Sold') {
        args.column.width = 60
    }
    },
    btnClick: function (args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.pdfExport({}, false, null, false, true);
    }
  },
  provide: {
    pivotview: [
      PDFExport,
      VirtualScroll
    ]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs272" %}

Similarly, you can change the height of specific rows in the PDF document by using the args.cell.height property in the onPdfCellRender event.

As shown in the code example below, the **“Mountain Bikes”** row under **“France”** is changed to a height of **30** pixels.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport"
        :enableVirtualization="enableVirtualization"
:onPdfCellRender="onPdfCellRender"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PDFExport, PdfCellRenderArgs, VirtualScroll } from
"@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      isPrimary: true,
      enableVirtualization: true,
    }
  },
```

```
  methods: {
    onPdfCellRender: function (args: PdfCellRenderArgs) {
      if (args.pivotCell && args.pivotCell.valueSort &&
args.pivotCell.valueSort.levelName === 'France.Mountain Bikes') {
        args.cell.height = 30
      }
    },
    btnClick: function (args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.pdfExport({}, false, null, false, true);
    }
  },
  provide: {
    pivotview: [
      PDFExport,
      VirtualScroll
    ]
  },
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs273" %}

This option is only available if enableVirtualization is set to **true**. In addition, the VirtualScroll and PDFExport modules must be injected into the pivot table.

Changing the pivot table style while exporting

The PDF export provides an option to change colors for headers, caption and records in pivot table before exporting. In-order to apply colors, define **theme** settings in **pdfExportProperties** object and pass it as a parameter to the pdfExport method.

By default, material theme is applied to exported PDF document.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let pdfExportProperties: PdfExportProperties = {
        theme: {
            header: {
                fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true, borders: { color: '#64FA50', lineStyle: 'Thin' }
            },
            record: {
                fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true
            },
            caption: {
                fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true
            }
        }
      };
      pivotGridObj.pdfExport(pdfExportProperties);
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs169" %}

<!-- markdownlint-disable MD009 -->

Changing default font while exporting

By default, the pivot table uses "Helvetica" font in the exported document. But it can be changed using the theme property in pdfExportProperties.

The available built-in fonts are,

- Helvetica
- TimesRoman
- Courier
- Symbol
- ZapfDingbats

`ts

import { PdfStandardFont, PdfFontFamily, PdfFontStyle } from '@syncfusion/ej2-pdf-export';

...

let pdfExportProperties: PdfExportProperties = {

theme: {

header: {font: new PdfStandardFont(PdfFontFamily.TimesRoman, 11, PdfFontStyle.Bold) },

caption: { font: new PdfStandardFont(PdfFontFamily.TimesRoman, 9) },

record: { font: new PdfStandardFont(PdfFontFamily.TimesRoman, 10) }

}

};

`

Adding custom font while exporting

In addition to existing built-in fonts, custom fonts can also be used. The custom font should be in **Base64** format and mention it in **PdfTrueTypeFont** class. In the following example, we have used **Advent Pro** font family that supports **Hungarian** language.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData, base64AlgeriaFont } from './pivotData.js';
import { PdfTrueTypeFont } from '@syncfusion/ej2-pdf-export';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
```

```
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let pdfExportProperties: PdfExportProperties = {
        theme: {
            header: {font:  new PdfTrueTypeFont(base64AlgeriaFont, 11) },
            caption: { font: new PdfTrueTypeFont(base64AlgeriaFont, 9) },
            record: { font: new PdfTrueTypeFont(base64AlgeriaFont, 10) }
        }
      };
      pivotGridObj.pdfExport(pdfExportProperties);
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs170" %}

The non-English alphabets can also be exported properly by setting its appropriate font.

Virtual Scroll Data

You can export the pivot table virtual scroll data as PDF document by using PivotEngine export without any performance degradation. To enable PivotEngine export in the pivot table, set the allowPdfExport as **true**. You need to use the exportToPDF method for PivotEngine export.

To use PivotEngine export, You need to inject the PDFExport module in pivot table.

PivotEngine export will be performed while enabling virtual scrolling by default

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PDFExport } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.pdfExportModule.exportToPDF();
    }
  },
  provide: {
    pivotview: [
      PDFExport
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs171" %}

Copyright © 2001 -2024 Syncfusion Inc.

Repeat row headers

Repeat row headers on each page can be achieved using PivotEngine export option. To disable repeat row headers, you need to set `allowRepeatHeader` to **false** in beforeExport event. You need to use the `exportToPDF` method for PivotEngine export.

To use PivotEngine export, You need to inject the `PDFExport` module in pivot table.

By default, repeat row headers is enabled in the PivotEngine export.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, PDFExport } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      isPrimary: true
    }
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.pdfExportModule.exportToPDF();
    }
  },
  provide: {
    pivotview: [
      PDFExport
```

```
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs172" %}

Export all pages

The pivot engine exports the entire virtual data of the pivot table (i.e. the data that contains all of the records used to render the complete pivot table) as a PDF document. To export just the current viewport of the pivot table, set the exportAllPages property to **false**. To use the pivot engine export, add the PDFExport module into the pivot table.

By default, the pivot engine export will be performed while virtual scrolling is enabled.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport"
:enableVirtualization="enableVirtualization"
exportAllPages="exportAllPages"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, VirtualScroll} from "@syncfusion/ej2-vue-
pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        rows: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        values: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      enableVirtualization: true,
```

```
      exportAllPages: true,
      isPrimary: true
    }
  },
  provide: {
    pivotview: [VirtualScroll]
  }
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.pdfExport();
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs173" %}

Events

PdfQueryCellInfo

The event pdfQueryCellInfo triggers on framing each row and value cell during PDF export. It allows the user to customize the cell value, style etc. of the current cell. It has the following parameters:

- value - It holds the cell value.
- column - It holds column information for the current cell.
- data - It holds the entire row data across the current cell.
- style - It holds the style properties for the cell.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"
:allowPdfExport="allowPdfExport"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      allowPdfExport: true,
      gridSettings: {
        columnWidth: 140,
        pdfQueryCellInfo: function(args){
            //triggers for every value cell while pdf export
        }
      }
    }
  }
  methods:{
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.pdfExport();
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs174" %}

PdfHeaderQueryCellInfo

The event pdfHeaderQueryCellInfo triggers on framing each column header cell during PDF export. It allows the user to customize the cell value, style etc. of the current cell. It has the following parameters:

- cell - It holds the current rendering cell information.
- style - It holds the style properties for the cell.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"
:allowPdfExport="allowPdfExport"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
```

```
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      allowPdfExport: true,
      gridSettings: {
        columnWidth: 140,
        pdfHeaderQueryCellInfo: function(args){
            //triggers for every header cell while pdf export
        }
      }
    }
  }
  methods:{
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.pdfExport();
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs175" %}

ExportComplete

The event exportComplete is triggered after the pivot table data has been exported to an PDF document. You can use this event to acquire blob stream data for further customization and processing at your end by passing the isBlob parameter as **true** when using the pdfExport method. It has the following parameters:

- type - It holds the current export type such as PDF, Excel, and CSV.
- promise - It holds the promise object for blob data.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="export-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">PDF Export</ejs-button>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <ejs-pivotview id="pivotview" :height="height"
:dataSourceSettings="dataSourceSettings" :allowPdfExport="allowPdfExport"
:enableVirtualization="enableVirtualization"
:exportComplete="exportComplete"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, VirtualScroll, ExportCompleteEventArgs, PDFExport
} from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        rows: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        values: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 320,
      allowPdfExport: true,
      enableVirtualization: true,
      isPrimary: true
    }
  },
  provide: {
    pivotview: [
      VirtualScroll,
      PDFExport
    ]
  },
  methods: {
    exportComplete: function (args: ExportCompleteEventArgs) {
      args.promise.then( function(e: { blobData: Blob }) {
        console.log(e.blobData);
      });
    },
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      let pdfExportProperties: PdfExportProperties = {
        fileName: 'pdfexport.csv'
      };
      pivotGridObj.pdfExport(pdfExportProperties, false, null, true);
    }
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs176" %}

See Also

- Excel Exporting

Globalization and localization in Vue Pivotview component

Globalization is the combination of Internationalization and localization. You can adapt the component to various languages by parsing and formatting the date or number (Internationalization) & adding culture specific customization and translation to the text (Localization).

Internationalization

Internationalization library provides support for formatting and parsing the number, date, and time by using the official Unicode CLDR JSON data and also provides the loadCldr method to load the culture specific CLDR JSON data.

By default, all the Essential JS 2 component are specific to English culture ('en-US'). If you want to go with the different culture other than English, follow the below steps.

- Install the CLDR-Data package by using the below command (it installs the CLDR JSON data). For more information about CLDR-Data, refer to this link.

`

npm install cldr-data --save

`

Once the package installed, you can find the culture specific JSON data under the location /node_modules/cldr-data.

- Now import the installed CLDR JSON data into the App.vue file. To import JSON data we need to install the JSON plugin loader. Here we have used the SystemJS JSON plugin loader.

`

npm install systemjs-plugin-json --save-dev

`

- Once installed, configure the system.config.js configuration settings as like below to map the

systemjs-plugin-json loader.

`ts

```
System.config({
paths: {
'syncfusion:': './node_modules/@syncfusion/'
},
map: {
app: 'app',
//Syncfusion packages mapping
"@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
"@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
"@syncfusion/ej2-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",
"@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",
"@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js",
"@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",
"@syncfusion/ej2-navigations": "syncfusion:ej2-navigations/dist/ej2-navigations.umd.min.js",
"@syncfusion/ej2-grids": "syncfusion:ej2-grids/dist/ej2-grids.umd.min.js",
"@syncfusion/ej2-dropdowns": "syncfusion:ej2-dropdowns/dist/ej2-dropdowns.umd.min.js",
"@syncfusion/ej2-calendars": "syncfusion:ej2-calendars/dist/ej2-calendars.umd.min.js",
"@syncfusion/ej2-lists": "syncfusion:ej2-lists/dist/ej2-lists.umd.min.js",
"@syncfusion/ej2-excel-export": "syncfusion:ej2-excel-export/dist/ej2-excel-export.umd.min.js",
"@syncfusion/ej2-pdf-export": "syncfusion:ej2-pdf-export/dist/ej2-pdf-export.umd.min.js",
"@syncfusion/ej2-file-utils": "syncfusion:ej2-file-utils/dist/ej2-file-utils.umd.min.js",
"@syncfusion/ej2-compression": "syncfusion:ej2-compression/dist/ej2-compression.umd.min.js",
"@syncfusion/ej2-pivotview": "syncfusion:ej2-pivotview/dist/ej2-pivotview.umd.min.js",
"@syncfusion/ej2-vue-base": "syncfusion:ej2-vue-base/dist/ej2-vue-base.umd.min.js",
"@syncfusion/ej2-vue-pivotview": "syncfusion:ej2-vue-pivotview/dist/ej2-vue-pivotview.umd.min.js",
"@syncfusion/ej2-vue-buttons": "syncfusion:ej2-vue-buttons/dist/ej2-vue-buttons.umd.min.js",
},
meta: {
'*.json': { loader: 'plugin-json' }
},
packages: {
'app': { main: 'app', defaultExtension: 'js' },
'cldr-data': { main: 'index.js', defaultExtension: 'js' }
```

Copyright © 2001 -2024 Syncfusion Inc.

}

});

System.import('app');

`

- Now import the required culture from the installed location to App.vue file, like the below code snippets.

`ts

//import the loadCldr from ej2-base

import { loadCldr} from '@syncfusion/ej2-base';

loadCldr(

require('cldr-data/supplemental/numberingSystems.json'),

require('cldr-data/main/fr-CH/ca-gregorian.json'),

require('cldr-data/main/fr-CH/numbers.json'));

require('cldr-data/main/fr-CH/timeZoneNames.json'));

`

The Internationalization library is used to globalize number, date, and time values in pivot table component using the dataSourceSettings.formatSettings option.

- Set the culture by using the locale property.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList" :locale="locale"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { loadCldr, L10n, setCulture, setCurrencyCode } from
'@syncfusion/ej2-base';
import * as currencies from './currencies.json';
import * as cagregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
import * as numberingSystems from './numberingSystems.json';
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
loadCldr(currencies, cagregorian, numbers, timeZoneNames, numberingSystems);
setCulture('de');
setCurrencyCode('EUR');
```

Copyright © 2001 -2024 Syncfusion Inc.

```
L10n.load({
    'de-DE': {
        'pivotview': {
            'grandTotal': 'Gesamtsumme',
            'total': 'Insgesamt',
            'value': 'Wert',
            'noValue': 'Kein Wert',
            'row': 'Zeile',
            'column': 'Spalte',
            'collapse': 'Zusammenbruch',
            'expand': 'Erweitern'
        },
        "pivotfieldlist": {
            'fieldList': 'Feld Liste',
            'dropRowPrompt': 'Drop Reihe hier',
            'dropColPrompt': 'Drop column Hier',
            'dropValPrompt': 'Drop wert hier',
            'dropFilterPrompt': 'Drop Filter Hier',
            'addPrompt': 'Feld hinzufügen',
            'centerHeader': 'Ziehen Sie die Felder zwischen den Bereichen
unten:',
            'add': 'Hinzufügen',
            'drag': 'Ziehen',
            'filters': 'Filter',
            'rows': 'Zeilen',
            'columns': 'Spalten',
            'values': 'Werte',
            'error': 'Fehler',
            'dropAction': 'Berechnetes Feld nicht in jeder anderen Region
außer Wert Achse sein.',
            'search': 'Suche',
            'close': 'Schließen',
            'cancel': 'Abbrechen',
            'delete': 'Löschen',
            'alert': 'Warnung',
            'warning': 'Warnung',
            'ok': 'OK',
            'allFields': 'Alle Felder',
            'noMatches': 'Keine Treffer'
        }
    }
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        filterSettings: [{ name: 'Country', type: 'exclude', items: ['United
States'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C2', useGrouping: false,
```

```
                minimumSignificantDigits: 1, maximumSignificantDigits: 3,
currency: 'EUR' }],
        filters: []
      },
      height: 350,
      showFieldList: true,
      locale: 'de-DE'
    }
  },
  provide: {
       pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs95" %}

* In the above sample, Amount field is formatted by NumberFormatOptions. For date formats, the value strings are formatted by DateFormatOptions.

* By default, locale value is en-US. If you want to change the en-US culture to a different culture, you have to change the locale accordingly.

* Also, you will find more details about support format string for number formats and data formats here.

<!-- markdownlint-disable MD009 -->

Decimal separators

The decimal separators of pivot table values varies based on the culture applied to the component. The culture can be set by calling the method setCulture with appropriate culture string as its parameter.

The following example demonstrates the decimal separators in Deutsch culture.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList" :locale="locale"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { loadCldr, L10n, setCulture, setCurrencyCode, Ajax } from
'@syncfusion/ej2-base';
import * as currencies from './currencies.json';
import * as cagregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
import * as numberingSystems from './numberingSystems.json';
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(PivotViewPlugin);
loadCldr(currencies, cagregorian, numbers, timeZoneNames, numberingSystems);
setCulture('de');
setCurrencyCode('EUR');
L10n.load({
    'de-DE': {
        'pivotview': {
            'grandTotal': 'Gesamtsumme',
            'total': 'Insgesamt',
            'value': 'Wert',
            'noValue': 'Kein Wert',
            'row': 'Zeile',
            'column': 'Spalte',
            'collapse': 'Zusammenbruch',
            'expand': 'Erweitern'
        },
        "pivotfieldlist": {
            'fieldList': 'Feld Liste',
            'dropRowPrompt': 'Drop Reihe hier',
            'dropColPrompt': 'Drop column Hier',
            'dropValPrompt': 'Drop wert hier',
            'dropFilterPrompt': 'Drop Filter Hier',
            'addPrompt': 'Feld hinzufügen',
            'centerHeader': 'Ziehen Sie die Felder zwischen den Bereichen
unten:',
            'add': 'Hinzufügen',
            'drag': 'Ziehen',
            'filters': 'Filter',
            'rows': 'Zeilen',
            'columns': 'Spalten',
            'values': 'Werte',
            'error': 'Fehler',
            'dropAction': 'Berechnetes Feld nicht in jeder anderen Region
außer Wert Achse sein.',
            'search': 'Suche',
            'close': 'Schließen',
            'cancel': 'Abbrechen',
            'delete': 'Löschen',
            'alert': 'Warnung',
            'warning': 'Warnung',
            'ok': 'OK',
            'allFields': 'Alle Felder',
            'noMatches': 'Keine Treffer'
        }
    }
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        filterSettings: [{ name: 'Country', type: 'exclude', items: ['United
States'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
```

```
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C2', currency: 'EUR' }],
        filters: []
      },
      height: 350,
      showFieldList: true,
      locale: 'de-DE'
    }
  },
  provide: {
        pivotview: [FieldList]
    },
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs96" %}

Localization

The Localization library allows you to localize default text content of the pivot table. The pivot table component has static text on some features (like drop area text, pivot field list title, etc...) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the

`locale` value and translation object.

The following list of properties and its values are used in the pivot table.

Locale keywords |Text

staticFieldList | Pivot Field List

fieldList | Field List

dropFilterPrompt | Drop filter here

dropColPrompt | Drop column here

dropRowPrompt | Drop row here

dropValPrompt | Drop value here

addPrompt | Add field here

adaptiveFieldHeader | Choose field

centerHeader | Drag fields between axes below:

add | add

drag | Drag

filter | Filter

filtered | Filtered

sort | Sort

Copyright © 2001 -2024 Syncfusion Inc.

remove | Remove

filters | Filters

rows | Rows

columns | Columns

values | Values

calculatedField | Calculated Field

createCalculatedField | Create Calculated Field

fieldName | Enter the field name

error | Error

invalidFormula | Invalid formula.

dropText | Example: ("Sum(Order*Count)" + "Sum(In*Stock)") * 250

dropTextMobile | Add fields and edit formula here.

dropAction | Calculated field cannot be place in any other region except value axis.

search | Search

close | Close

cancel | Cancel

delete | Delete

alert | Alert

warning | Warning

ok | OK

sum | Sum

average | Average

count | Count

min | Min

max | Max

allFields | All Fields

formula | Formula

fieldExist | A field already exists in this name. Please enter a different name.

confirmText | A calculation field already exists in this name. Do you want to replace it?

noMatches | No matches

format | Summaries values by

edit | Edit

clear | Clear

Copyright © 2001 -2024 Syncfusion Inc.

formulaField | Drag and drop fields to formula

dragField | Drag field to formula

Loading Translations

To load translation object in an application, use load function of the L10n class.

The following example demonstrates the pivot table in Deutsch culture.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList" :locale="locale"> </ejs-
pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { L10n } from '@syncfusion/ej2-base';
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
L10n.load({
    'de-DE': {
        'pivotview': {
            'grandTotal': 'Gesamtsumme',
            'total': 'Insgesamt',
            'value': 'Wert',
            'noValue': 'Kein Wert',
            'row': 'Zeile',
            'column': 'Spalte',
            'collapse': 'Zusammenbruch',
            'expand': 'Erweitern'
        },
        "pivotfieldlist": {
            'fieldList': 'Feld Liste',
            'dropRowPrompt': 'Drop Reihe hier',
            'dropColPrompt': 'Drop column Hier',
            'dropValPrompt': 'Drop wert hier',
            'dropFilterPrompt': 'Drop Filter Hier',
            'addPrompt': 'Feld hinzufügen',
            'centerHeader': 'Ziehen Sie die Felder zwischen den Bereichen
unten:',
            'add': 'Hinzufügen',
            'drag': 'Ziehen',
            'filters': 'Filter',
            'rows': 'Zeilen',
            'columns': 'Spalten',
            'values': 'Werte',
            'error': 'Fehler',
            'dropAction': 'Berechnetes Feld nicht in jeder anderen Region
außer Wert Achse sein.',
            'search': 'Suche',
            'close': 'Schließen',
            'cancel': 'Abbrechen',
            'delete': 'Löschen',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            'alert': 'Warnung',
            'warning': 'Warnung',
            'ok': 'OK',
            'allFields': 'Alle Felder',
            'noMatches': 'Keine Treffer'
        }
    }
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        filterSettings: [{ name: 'Country', type: 'exclude', items: ['United
States'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true,
      locale: 'de-DE'
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs97" %}

Right-to-left (RTL)

RTL provides an option to switch the text direction and layout of the pivot table component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc...). To enable RTL pivot table, set the `enableRtl` property to **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList" :locale="locale"
:enableRtl="enableRtl"> </ejs-pivotview>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { L10n } from '@syncfusion/ej2-base';
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
L10n.load({
    'ar-AE': {
        'pivotview': {
            'grandTotal': 'المجموع الكلى',
            'total': 'المجموع',
            'value': 'القيمة',
            'noValue': 'لا قيمة لها',
            'row': 'صف',
            'column': 'العمود',
            'collapse': 'الانهيار',
            'expand': 'توسيع'
        },
        'pivotfieldlist': {
            'fieldList': 'قائمة الحقول',
            'dropRowPrompt': 'تراجع الخلاف هنا',
            'dropColPrompt': 'انخفاض العمود هنا',
            'dropValPrompt': 'انخفاض قيمة هنا',
            'dropFilterPrompt': 'انخفاض هنا عامل التصفية',
            'addPrompt': 'اضافة حقل هنا',
            'adaptiveFieldHeader': 'اختر الحقل',
            'centerHeader': 'اسحب المجالات بين المناطق الموضحة ادناه:',
            'add': 'اضافة',
            'drag': 'اسحب',
            'filters': 'عوامل التصفية',
            'rows': 'الصفوف',
            'columns': 'الاعمدة',
            'values': 'قيم',
            'search': 'البحث',
            'close': 'قريب',
            'cancel': 'الغاء',
            'delete': 'احذف',
            'alert': 'حالة تاهب قصوى',
            'warning': 'تحذير',
            'ok': 'موافق',
            'allFields': 'جميع الحقول',
            'noMatches': 'لا مباريات'
        }
    }
});
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        filterSettings: [{ name: 'Country', type: 'exclude', items: ['United
States'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
```

```
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true,
      locale: 'ar-AE',
      enableRtl: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs98" %}

See Also

- Internationalization
- Localization

Accessibility in Vue Pivotview component

The pivot table component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the pivot table component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/not-supported.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

WAI-ARIA (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript,and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components. The following ARIA attributes are used in the pivot table component:

| Attributes | Purpose |

| --- | --- |

| `role=grid` | Attribute added to identify the grid component element within the pivot table element. |

| `role=region` | Attribute added to identify the chart component element within the pivot table element. |

| `role=button` | This attribute is added to the pager navigation buttons as well as the buttons in the dialog popup such as field list, calculated field, member editor, conditional formatting of pivot table component to indicate that it is a clickable element. |

| `role=table` | This attribute is added to each conditional formatting style container element to denote it as a table. |

| `role=tableItems` | This attribute is added to the container element that appears inside the number formatting popup to indicate it as a table. |

| `aria-disabled` | The buttons within the dialog popups, such as field list, calculated field and member editor, will be disabled based on their usability. To indicate its disabled state, we will add this attribute with the values `true`. By default, the attribute value is set to `false`. |

| `aria-label` | This attribute is added to label elements that are placed inside the pager, member editor popup, and calculated field popup to identify them as label elements. |

Copyright © 2001 -2024 Syncfusion Inc.

| aria-selected | This attribute is added to the selected treeview item in the calculated field popup with the value as true to denote that it is a selected element. |

| aria-colspan | This attribute is added to the th elements in the e-table, which represent the column span value. |

| aria-rowspan | This attribute is added to the th elements in the e-table, which represent the row span value. |

| data-type | This attribute is added to the treeview item in the calculated field popup, as well as the buttons in the grouping bar and field list. It represents the aggregate type for the specified field. |

| data-caption | This attribute is added to the treeview item in the calculated field popup, as well as the buttons in the grouping bar and field list. It represents the caption for the specified field. |

| data-basefield | This attribute is added to the treeview item in the calculated field popup, as well as the buttons in the grouping bar and field list. It denotes the base field for the specified field, which is used to display the values for aggregation types such as **DifferenceFrom**, **PercentageOfDifferenceFrom**, and **PercentageOfParentTotal**. |

| data-baseitem | This attribute is added to the treeview item in the calculated field popup, as well as the buttons in the grouping bar and field list. It denotes the base item for the specified field, which is used to display the values for aggregation types such as **DifferenceFrom**, **PercentageOfDifferenceFrom**, and **PercentageOfParentTotal**. |

| data-field | This attribute is added to the treeview item in the calculated field popup. It denotes the name of the specified field. |

| data-membertype | This attribute is added to the treeview item in the calculated field popup. It denotes the member type of the selected OLAP calculated field. |

| data-hierarchy | This attribute is added to the treeview item in the calculated field popup. It denotes the parent hierarchy unique name of the selected OLAP calculated field. |

| data-formula | This attribute is added to the treeview item in the calculated field popup. It denotes the formula used for the specified calculated field. |

| data-formatString | This attribute is added to the treeview item in the calculated field popup. It denotes the format string used for the specified calculated field. |

| data-customformatstring | This attribute is added to the treeview item in the calculated field popup. It denotes the custom format string used for the specified calculated field. |

Keyboard interaction

The pivot table component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Message component.

| **Press** | **To do this** |

| --- | --- |

| Tab / Shift + Tab | To focus the close icon in the message. |

| Enter / Space | Closes the focused close icon's message. |

Copyright © 2001 -2024 Syncfusion Inc.

Pivot Table

| **Press** | **To do this** |

| --- | --- |

| Tab | Moves the cell focus right side. If no cells are focused, it moves to the next active element in the browser page. |

| Shift + Tab | Moves the cell focus left side. If no cells are focused, it moves to the previous active element in the browser page. |

| DownArrow | Moves the cell focus downwards. If the selection is enabled in the pivot table, then it will move downwards to select next row or column or individual cell. |

| UpArrow | Moves the cell focus upwards. If the selection is enabled in the pivot table, then it will move upwards to select previous row or column or individual cell. |

| LeftArrow | Moves the cell focus left side. If the selection is enabled in the pivot table, then it will move left side to select previous row or column or individual cell. |

| RightArrow | Moves the cell focus right side. If the selection is enabled in the pivot table, then it will move right side to select next row or column or individual cell. |

| Shift + DownArrow | Extends the cell selection downwards. |

| Shift + UpArrow | Extends the cell selection selection upwards. |

| Shift + LeftArrow | Extends the cell selection to the left side. |

| Shift + RightArrow | Extends the cell selection to the right side. |

| Ctrl + A | Selects all cells. |

| Esc | Deselects all cells. If the current active element is a context menu, then the context menu popup will be closed. |

| Home | Goes to the first cell in the current row. |

| End | Goes to the last cell in the current row. |

| Ctrl + Home | Goes to the first cell in the table. |

| Ctrl + End | Goes to the last cell in the table. |

| Enter | If the current cell is an expand/collapse cell, it performs expand/collapse operation (drill operation). If the current row/column header is in value sort state, it performs value sorting. If the current cell is in selection state, it moves to the next row, column or individual cell. If drill-through or editing is enabled in the pivot table, the drill-through dialog will be opened based on the selected value cell. If the current active element is a context menu popup, menu selection will be performed. |

| Shift + Enter | If value sorting is enabled in the pivot table and the current cell is a header with respect to its value axis, it performs value sorting to either ascending or descending order. If the current cell is in selection state, it moves to the previous row, column or individual cell. |

| Ctrl + Enter | If hyperlink is enabled in the current cell, it performs hyperlink selection. |

Copyright © 2001 -2024 Syncfusion Inc.

| Shift + F10 or Menu | If context menu is enabled in the pivot table, the context menu popup will be opened in the current cell. |

Field List

| **Press** | **To do this** |

| --- | --- |

| Shift + Ctrl + F | If the popup field list is enabled in either the pivot table or the pivot chart, the field list dialog will be opened. |

| Tab | Moves to the next active element in the field list. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the field list. If no active elements present, it moves to the previous active element in the browser page. |

| Shift + F | If the current active element is a field's button and if it has a filter icon, the filter dialog will open to perform filtering. |

| Shift + S | If the current active element is a field's button and if it has a sort icon, the sorting will be performed to the selected field. |

| Shift + E | If the current active element is a calculated field's button and if it has an edit icon, the calculated field dialog will be opened to perform editing the selected calculated field. |

| Enter | Performs the selection operation of the current active element. If the current active element is a field's button and it has a dropdown icon, the aggregation menu will open to perform calculations using aggregation options to the selected value field. |

| Delete | If the current active element is a field's button, the selected field will be removed from the current report. |

| DownArrow | If the current active element is a tree node, it moves to the next node. |

| UpArrow | If the current active element is a tree node, it moves to the previous node. |

| LeftArrow | If the current active element is a tree node, it collapses the current node. |

| RightArrow | If the current active element is a tree node, it expands the current node. |

| Home | If the current active element is a tree node, it goes to the first node. |

| End | If the current active element is a tree node, it goes to the last node. |

| Space | If the current active element is a tree node or a checkbox element, it will be either checked or unchecked. |

| Esc or Escape | Closes the popup field list dialog. |

Grouping Bar

| **Press** | **To do this** |

| --- | --- |

| Tab | Moves to the next active element (field's button) in the grouping bar. If no active elements present, it moves to the next active element in the browser page. |

Copyright © 2001 -2024 Syncfusion Inc.

| Shift + Tab | Moves to the previous active element (field's button) in the grouping bar. If no active elements present, it moves to the previous active element in the browser page. |

| Shift + F | If the current active element is a field's button and if it has a filter icon, the filter dialog will be opened to perform filtering. |

| Shift + S | If the current active element is a field's button and if it has a sort icon, the sorting will be performed to the selected field. |

| Shift + E | If the current active element is a calculated field's button and if it has an edit icon, the calculated field dialog will be opened to perform editing the selected calculated field. |

| Enter | Performs the selection operation of the current active element. If the current active element is a field's button and if it has a dropdown icon, the aggregation menu will be opened to perform calculations using aggregation options to the selected value field. |

| Delete | If the current active element is a field's button, the selected field will be removed from the current report. |

| DownArrow | If the current active element is a dropdown list, the next item will be selected. |

| UpArrow | If the current active element is a dropdown list, the previous item will be selected. |

| Home | If the current active element is a dropdown list, the first item will be selected. |

| End | If the current active element is a dropdown list, the last item will be selected. |

| Alt + Down | If the current active element is a dropdown list, the popup will be opened. |

| Alt + Down | If the current active element is a dropdown list, the popup will be closed. |

| Esc or Escape | Closes the dropdown list.

Filter Dialog

| **Press** | **To do this** |

| --- | --- |

| Shift + F | If the current active element is a field's button and if it has a filter icon in either the field list or grouping bar UI, the filter dialog will be opened to perform filtering. |

| Tab | Moves to the next active element in the filter dialog. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the filter dialog. If no active elements present, it moves to the previous active element in the browser page. |

| DownArrow | If the current active element is a tree node, it moves to the next node. |

| UpArrow | If the current active element is a tree node, it moves to the previous node. |

| LeftArrow | If the current active element is a tree node, it collapses the current node. If the current active element is a tab, it moves focus to the previous tab element. |

Copyright © 2001 -2024 Syncfusion Inc.

| RightArrow | If the current active element is a tree node, it expands the current node. If the current active element is a tab, it moves focus to the next tab element. |

| Home | If the current active element is a tree node, it goes to the first node. |

| End | If the current active element is a tree node, it goes to the last node. |

| Space | If the current active element is a tree node or a checkbox element, it will be either checked or unchecked. |

| Alt + Down | If the current active element is a DropDownList or DatePicker or DateTimePicker, the popup will be opened. |

| Alt + Up | If the current active element is a DropDownList or DatePicker or DateTimePicker, the popup will be closed. |

| Enter | Performs the selection operation of the current active element. If the current active element is a tab, the current tab element will be selected. If the current active element is a tree node, the current node will be either checked or unchecked. If the current active element is DropDownList, the focus item will be selected, and the popup list will close when it is open. Otherwise, toggles the popup list. |

| Esc or Escape | Closes the filter dialog. |

Calculated Field Dialog

| **Press** | **To do this** |

| --- | --- |

| Shift + E | If the current active element is a field's button and if it has an edit icon in either the field list or grouping bar UI, the calculated field dialog will be opened to perform editing the selected calculated field. |

| Tab | Moves to the next active element in the calculated field dialog. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the calculated field dialog. If no active elements present, it moves to the previous active element in the browser page. |

| DownArrow | If the current active element is a tree node, it moves to the next node. |

| UpArrow | If the current active element is a tree node, it moves to the previous node. |

| LeftArrow | If the current active element is a tree node, it collapses the current node. |

| RightArrow | If the current active element is a tree node, it expands the current node. If the current active element is a tree node and has a menu icon, the aggregation menu will be opened to select appropriate aggregation type to the selected field. |

| Home | If the current active element is a tree node, it goes to the first node. |

| End | If the current active element is a tree node, it goes to the last node. |

Copyright © 2001 -2024 Syncfusion Inc.

| Enter | Performs the selection operation of the current active element. If the current active element is a tree node, it copies the selected field name/formula to the formula text area to perform calculations. |

| Esc or Escape | Closes the calculated field dialog. |

Formatting Dialog

| **Press** | **To do this** |

| --- | --- |

| Tab | Moves to the next active element in the formatting dialog. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the formatting dialog. If no active elements present, it moves to the previous active element in the browser page. |

| DownArrow | If the current active element is a DropDownList, the next item will be selected. |

| UpArrow | If the current active element is a DropDownList, the previous item will be selected. |

| Home | If the current active element is a DropDownList, the first item will be selected. |

| End | If the current active element is a DropDownList, the last item will be selected. |

| Alt + Down | If the current active element is a DropDownList or ColorPicker, the popup will be opened. |

| Alt + Down | If the current active element is a DropDownList or ColorPicker, the popup will be closed. |

| Enter | Performs the selection operation of the current active element. |

| Esc or Escape | Closes the formatting dialog. |

Toolbar

| **Press** | **To do this** |

| --- | --- |

| Tab | Moves to the next active option in the toolbar. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active option in the toolbar. If no active elements present, it moves to the previous active element in the browser page. |

| Enter | Performs the selection operation of the current active element. |

Drill-Through Dialog

| **Press** | **To do this** |

| --- | --- |

Copyright © 2001 -2024 Syncfusion Inc.

| Tab | Moves to the next active element in the drill-through dialog. If the current active element is a Grid cell, it moves the cell focus to right side. If no active elements present, then it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the drill-through dialog. If the current active element is a Grid cell, it moves the cell focus to left side, If no active elements present, then it moves to the previous active element in the browser page. |

| DownArrow | Moves the row/cell focus downwards. |

| UpArrow | Moves the row/cell focus upwards. |

| LeftArrow | Moves the cell focus left side. |

| RightArrow | Moves the cell focus right side. |

| Home | Goes to the first cell in the current row. |

| End | Goes to the last cell in the current row. |

| Ctrl + Home | Goes to the first cell in the table. |

| Ctrl + End | Goes to the last cell in the table. |

| Enter | Performs the selection operation of the current active element. |

| Esc or Escape | If the cell is in selected state, the it deselects all rows/cells. If the row/cell is in edit state, it cancels the current entries in the row/cell. If the current active element is not a row/cell, it closes the drill-through dialog. |

| F2 | Initiate editing a row/cell in the data grid. |

| Insert | Adds a new row/cell in the data grid. |

| Delete | Removes the selected row in the data grid. |

Some commonly used applicable key combinations and their relative functionalities in all dialogs are listed below.

| **Press** | **To do this** |

| --- | --- |

| Tab | Moves to the next active element in the dialog. If either no active elements present in the dialog or an overlay is not present in the dialog, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the dialog. If either no active elements present in the dialog or an overlay is not present in the dialog, it moves to the previous active element in the browser page. |

| Space | If the current active element is a tree node or a checkbox element, it will be either checked or unchecked. |

| Enter | When the Dialog button or any input (except text area) is in focus state, when pressing the Enter key, the click event associated with the primary button or button will be triggered.

Copyright © 2001 -2024 Syncfusion Inc.

The Enter key will not be worked, when the dialog content contains any text area with initial focus. |

| Esc or Escape | Closes the dialog. |

Ensuring accessibility

The pivot table component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the pivot table component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the pivot table component with accessibility tools.

~/SRC/APP.VUE

```
<template>
    <div id="app">
      <ejs-pivotview id="pivotview" ref="pivotview"
        :dataSourceSettings="dataSourceSettings"
:gridSettings="gridSettings" :height="height" :showFieldList="showFieldList"
:allowCalculatedField="allowCalculatedField"
        :allowExcelExport="allowExcelExport"
:allowPdfExport="allowPdfExport" :showToolbar="showToolbar"
:allowConditionalFormatting="allowConditionalFormatting"
        :allowNumberFormatting="allowNumberFormatting"
:showGroupingBar="showGroupingBar"
:groupingBarSettings="groupingBarSettings"
        :toolbar="toolbar" :saveReport="saveReport" :loadReport="loadReport"
:fetchReport="fetchReport" :renameReport="renameReport"
        :removeReport="removeReport" :newReport="newReport"
:displayOption="displayOption">
      </ejs-pivotview>
  </div>
</template>
<script>
  import { PivotViewComponent, FieldList, CalculatedField, Toolbar,
PDFExport, ExcelExport, ConditionalFormatting } from "@syncfusion/ej2-vue-
pivotview";
  import { GroupingBar, Grouping, NumberFormatting, DrillThrough } from
"@syncfusion/ej2-vue-pivotview";
  import { pivot_dataset } from './pivotData.js';
  export default {
    components: {
      'ejs-pivotview': PivotViewComponent
    },
    data () {
      return {
        dataSourceSettings: {
          dataSource: pivot_dataset,
          expandAll: true,
          enableSorting: true,
          allowLabelFilter: true,
          allowValueFilter: true,
          sortSettings: [
            { name: 'company', order: 'Descending' }
          ],
          formatSettings: [
```

```
            { name: 'balance', format: 'C' },
            { name: 'date', format: 'dd/MM/yyyy-hh:mm', type: 'date' }
          ],
          drilledMembers: [
            { name: 'product', items: ['Bike', 'Car'] },
            { name: 'gender', items: ['male'] }
          ],
          filterSettings: [
            { name: 'date', type: 'Date', condition: 'Between', value1: new
Date('02/16/2000'), value2: new Date('02/16/2002') },
            { name: 'age', type: 'Number', condition: 'Between', value1:
'25', value2: '35' },
            { name: 'eyeColor', type: 'Exclude', items: ['blue'] }
          ],
          rows: [
            { name: 'state' },
            { name: 'eyeColor' }
          ],
          columns: [
            { name: 'gender', caption: 'Population' },
            { name: 'isActive' }
          ],
          values: [
            { name: 'balance' },
            { name: 'quantity' }
          ],
          filters: [],
          conditionalFormatSettings: [
            {
              measure: 'balance',
              value1: 100000,
              conditions: 'LessThan',
              style: {
                backgroundColor: '#80cbc4',
                color: 'black',
                fontFamily: 'Tahoma',
                fontSize: '12px'
              }
            },
            {
              value1: 10,
              value2: 20,
              measure: 'quantity',
              conditions: 'Between',
              style: {
                backgroundColor: '#f48fb1',
                color: 'black',
                fontFamily: 'Tahoma',
                fontSize: '12px'
              }
            }
          ]
        },
        showGroupingBar: true,
          groupingBarSettings: {
            showFieldsPanel: true
          },
```

```
            displayOption: { view: 'Both' },
            chartSettings: {
              value: 'Amount', enableExport: true,
              chartSeries: {
                type: 'Column',
                animation: {
                  enable: false
                }
              },
              enableMultipleAxis: false,
            },
            toolbar: [
              'New', 'Save', 'SaveAs', 'Rename', 'Remove', 'Load', 'Grid',
'Chart', 'MDX', 'Export', 'SubTotal', 'GrandTotal', 'Formatting',
'FieldList'
            ],
            editSettings: { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Normal' },
            allowExcelExport: true,
            allowConditionalFormatting: true,
            allowNumberFormatting: true,
            allowPdfExport: true,
            allowGrouping: true,
            showToolbar: true,
            allowCalculatedField: true,
            showFieldList: true,
            allowDeferLayoutUpdate: true,
            saveReport: function (args) {
              var reports = [];
              var isSaved = false;
              if (localStorage.pivotviewReports &&
localStorage.pivotviewReports !== "") {
                reports = JSON.parse(localStorage.pivotviewReports);
              }
              if (args.report && args.reportName && args.reportName !== '') {
                reports.map(function (item) {
                if (args.reportName === item.reportName) {
                  item.report = args.report;
                  isSaved = true;
                }
                });
                if (!isSaved) {
                  reports.push(args);
                }
                localStorage.pivotviewReports = JSON.stringify(reports);
              }
            },
            fetchReport: function (args) {
              var reportCollection = [];
              var reeportList = [];
              if (localStorage.pivotviewReports &&
localStorage.pivotviewReports !== "") {
                reportCollection = JSON.parse(localStorage.pivotviewReports);
              }
              reportCollection.map(function (item) {
                reeportList.push(item.reportName);
              });
```

```
            args.reportName = reeportList;
          },
          loadReport: function (args) {
            var reportCollection = [];
            if (localStorage.pivotviewReports &&
localStorage.pivotviewReports !== "") {
              reportCollection = JSON.parse(localStorage.pivotviewReports);
            }
            reportCollection.map(function (item) {
                if (args.reportName === item.reportName) {
                  args.report = item.report;
                }
            });
            if (args.report) {
              pivotGridObj.dataSourceSettings =
JSON.parse(args.report).dataSourceSettings;
            }
          },
          removeReport: function (args) {
            var reportCollection = [];
            if (localStorage.pivotviewReports &&
localStorage.pivotviewReports !== "") {
              reportCollection = JSON.parse(localStorage.pivotviewReports);
            }
            for (var i = 0; i < reportCollection.length; i++) {
              if (reportCollection[i].reportName === args.reportName) {
                reportCollection.splice(i, 1);
              }
            }
            if (localStorage.pivotviewReports &&
localStorage.pivotviewReports !== "") {
              localStorage.pivotviewReports =
JSON.stringify(reportCollection);
            }
          },
          renameReport: function (args) {
            var reportCollection = [];
            if (localStorage.pivotviewReports &&
localStorage.pivotviewReports !== "") {
              reportCollection = JSON.parse(localStorage.pivotviewReports);
            }
            if (args.isReportExists) {
              for (var i = 0; i < reportCollection.length; i++) {
                if (reportCollection[i].reportName === args.rename) {
                  reportCollection.splice(i, 1);
                }
              }
            }
            reportCollection.map(function (item) {
              if (args.reportName === item.reportName) {
                item.reportName = args.rename;
              }
            });
            if (localStorage.pivotviewReports &&
localStorage.pivotviewReports !== "") {
              localStorage.pivotviewReports =
JSON.stringify(reportCollection);
```

```
            }
          },
          newReport: function () {
            pivotGridObj.setProperties({
              dataSourceSettings: {
                columns: [],
                rows: [],
                values: [],
                filters: []
              }
            }, false);
          },
        toolbarRender: function (args) {
          args.customToolbar.splice(6, 0, {
            type: 'Separator'
          });
          args.customToolbar.splice(9, 0, {
            type: 'Separator'
          });
        },
        gridSettings: {
          columnWidth: 140,
          contextMenuItems: [
            'Aggregate', 'CalculatedField', 'Drillthrough', 'Excel Export',
'Pdf Export', 'Csv Export', 'Expand', 'Collapse', 'Sort Ascending', 'Sort
Descending'
          ]
        },
        width: '100%',
        height: 450
      }
    },
    provide: {
      pivotview: [
        FieldList, CalculatedField, Toolbar, PDFExport, ExcelExport,
ConditionalFormatting, GroupingBar, GroupingBar, Grouping, NumberFormatting,
DrillThrough
      ]
    }
  }
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/accessibility" %}

Copyright © 2001 -2024 Syncfusion Inc.

See also

- Accessibility in Syncfusion Vue components

How To

<!-- markdownlint-disable MD009 -->

Switching older themes style in Vue Pivotview component

From Volume 1, 2020 onwards Syncfusion has revised the theming and layout of the Pivot Table. So, to inherit the older theme style and layout please do the necessary changes in CSS and pivot table height.

CSS Selectors

In current theme, the cells can be differentiated by their background colors. To avoid it, you need to override its background colors via simple CSS coding within your application. The below CSS selectors allow to achieve the same for material, fabric, bootstrap and bootstrap v4 themes.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<!-- Codes here... -->
<style>
.e-pivotview .e-rowsheader,
.e-pivotview .e-columnsheader,
.e-pivotview .e-gtot,
.e-pivotview .e-content,
.e-pivotview .e-gridheader,
.e-pivotview .e-headercell {
background-color:#fff !important;
}
</style>
</head>
<body>
</body>
</html>
```

Meanwhile for high contrast theme, we need to set the following CSS.

```
<!DOCTYPE html>
```

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<!-- Codes here... -->

<style>

.e-pivotview .e-rowsheader,

.e-pivotview .e-columnsheader,

.e-pivotview .e-gtot,

.e-pivotview .e-content,

.e-pivotview .e-gridheader,

.e-pivotview .e-headercell {

background-color:#000 !important;

}

</style>

</head>

<body>

</body>

</html>

`

Adjusting Row Height

In current theme, to make the component compact we have reduced the height of each pivot table rows. But user can reset the height of the pivot table using the rowHeight property in gridSettings. In older theme, the property was set to 36 pixels for desktop layout and 48 pixels for mobile layout. So reset the rowHeight accordingly to visualize the older theme style.

In the below code sample, we replicate the older theme style.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
```

```
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: {
          rowHeight: 36
      }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
/* csslint ignore:start */
    .e-pivotview .e-rowsheader,
    .e-pivotview .e-columnsheader,
    .e-pivotview .e-gtot,
    .e-pivotview .e-content,
    .e-pivotview .e-gridheader,
    .e-pivotview .e-headercell {
        background-color:#fff !important;
    }
/* csslint ignore:end */
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs142" %}

Customize number date and time values in Vue Pivotview component

You can format the number, date, and time values for each field using formatSettings option under dataSourceSettings. It can be configured through code behind, during initial rendering.

Number formatting

For numbers, the formatting settings required to apply through code behind are:

- name: It allows to set the field name.
- format: It allows to set the format of the respective field.

Also, you can customize the applied number format by setting the NumberFormatOptions options in formatSettings itself.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
```

```
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        formatSettings: [{ name: 'Amount', format: 'C2', useGrouping: false,
                minimumSignificantDigits: 1, maximumSignificantDigits: 3 }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs135" %}

Date and Time formatting

For date and time, the formatting settings required to apply through code behind are:

- name: It allows to set the field name.
- format: It allows to set the format of the respective field.
- type: It allows to set the type of format to be used for the respective field.

Also, you can customize the applied date format by setting DateFormatOptions options in formatSettings itself.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
```

```
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        formatSettings: [{ name: 'Year', format: 'dd/MM/yyyy-hh:mm', type:
'date' }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        filters: []
      },
      height: 350
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs136" %}

Limitations of date formatting

As per Firefox and Edge browsers standards, most of the date and time formats used in data source aren't supported. For example: Apr-2000, Apr-01-2000, 01-03-2000, 2000-Apr-01 etc... are not supported. Meanwhile ISO formats will be supported across all browsers.

Customize the icons for pivot grid in Vue Pivotview component

You can customize the pivot button icons in the pivot table by overriding the class **.pivot-button** with a custom property content as mentioned below.

`

PivotView_PivotFieldList .e-icons.e-toggle-field-list::before {
content: '\e337';

}

`

In the below sample, pivot table is rendered with a customized pivot button icons.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
:showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
```

```
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: [],
        calculatedFieldSettings: [{ name: 'Total', formula:
'"Sum(Amount)"+"Sum(Sold)"' }]
      },
      showFieldList: true,
      height: 350
    }
  },
  provide: {
    pivotview: [FieldList]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
/* csslint ignore:start */
#PivotView_PivotFieldList_Wrapper .e-pivot-button .e-icons.e-dropdown-
icon::before,
#PivotView .e-pivot-button .e-icons.e-dropdown-icon::before {
    content: "\e941";
}
#PivotView_PivotFieldList_Wrapper .e-pivot-button .e-icons.e-pv-
filtered::before,
#PivotView .e-pivot-button .e-icons.e-pv-filtered::before {
    content: "\e974";
}
#PivotView_PivotFieldList_Wrapper .e-pivot-button .e-icons.e-pv-
filter::before,
#PivotView .e-pivot-button .e-icons.e-pv-filter::before {
    content: '\e202';
}
#PivotView_PivotFieldList .e-icons.e-toggle-field-list::before {
    content: '\e337';
}
#PivotView_PivotFieldList_Wrapper .e-pivot-button .e-icons.e-sort::before,
#PivotView .e-pivot-button .e-icons.e-sort::before {
    content: '\e306';
}
#PivotView_PivotFieldList_Wrapper .e-pivot-button .e-icons.e-remove::before,
#PivotView .e-pivot-button .e-icons.e-remove::before {
```

```
    content: '\e201';
}
/* csslint ignore:end */
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs137" %}

Configure data grid options on editing mode in Vue Pivotview component

You can access the data grid options such as sort, group, filter on editing mode using the beginDrillThrough event in the pivot table. The event occurs in every value cell on double click and provides the data grid information before display the drill through grid pop-up.

Grid features are segregated into individual feature-wise modules. For example, to use sorting feature, you should inject Sort using the Grid.Inject(Sort) section.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :editSettings="editSettings"
:beginDrillThrough="beginDrillThrough"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, BeginDrillThroughEventArgs } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
import { Grid, Sort, Filter, Group, Inject } from '@syncfusion/ej2-grids';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      editSettings: { allowAdding: true, allowDeleting: true, allowEditing:
true, mode: 'Normal' }
    }
  },
  methods: {
    beginDrillThrough: function(args: BeginDrillThroughEventArgs) {
        if (args.gridObj) {
            Grid.Inject(Sort, Filter, Group);
            let gridObj: Grid = args.gridObj;
```

```
            gridObj.allowGrouping = true;
            gridObj.allowSorting = true;
            gridObj.allowFiltering = true;
            gridObj.filterSettings = { type: 'CheckBox' };
        }
    },
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs133" %}

Refresh the field list in Vue Pivotview component

You can refresh pivot table and field list with new data source dynamically.

APP.VUE

```
<template>
    <div id="app">
        <ejs-button id="refresh-btn" :isPrimary="isPrimary" v-
on:click.native="btnClick">Refresh</ejs-button>
        <ejs-pivotview id="pivotview" :height="height"
:showFieldList="showFieldList" :dataSourceSettings="dataSourceSettings" >
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { ButtonPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-buttons";
Vue.use(PivotViewPlugin);
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: [
            { 'Sold': 31, 'Amount': 52824, 'Country': 'France', 'Products':
'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q1' },
            { 'Sold': 51, 'Amount': 86904, 'Country': 'France', 'Products':
'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q2' },
            { 'Sold': 90, 'Amount': 153360, 'Country': 'France', 'Products':
'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q3' },
            { 'Sold': 25, 'Amount': 42600, 'Country': 'France', 'Products':
'Mountain Bikes', 'Year': 'FY 2015', 'Quarter': 'Q4' },
            { 'Sold': 27, 'Amount': 46008, 'Country': 'France', 'Products':
'Mountain Bikes', 'Year': 'FY 2016', 'Quarter': 'Q1' }],
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
```

```
        filters: []
      },
      showFieldList: true,
      height: 350,
      isPrimary: true
    }
  },
  provide: {
    pivotview: [FieldList]
  },
  methods: {
    btnClick: function(args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.engineModule.fieldList = {};
      pivotGridObj.dataSourceSettings.dataSource = [
            { 'Sold': 31, 'Amount': 52824, 'Country': 'France', 'Year': 'FY
2015' },
            { 'Sold': 51, 'Amount': 86904, 'Country': 'France', 'Year': 'FY
2015' },
            { 'Sold': 90, 'Amount': 153360, 'Country': 'France', 'Year': 'FY
2015' },
            { 'Sold': 25, 'Amount': 42600, 'Country': 'France', 'Year': 'FY
2015' },
            { 'Sold': 27, 'Amount': 46008, 'Country': 'France', 'Year': 'FY
2016' }];
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs12" %}

Hide empty headers in Vue Pivotview component

If the raw data for a particular field is not defined, it will be shown as 'Undefined' in the pivot table headers. You can hide those headers by setting the showHeaderWhenEmpty property to **false** in the pivot table.

For example, if the raw data for the field 'Country' is defined as **“United Kingdom”** and **“State”** is not defined means, it will be shown as **“United Kingdom >> Undefined”** in the header section. Here, you can hide those 'Undefined' header using the showHeaderWhenEmpty property.

By default, this property is set as **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings" :height=
"height" :showFieldList="showFieldList"> </ejs-pivotview>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotNullData } from './pivotNullData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotNullData,
        expandAll: false,
        rows: [{ name: 'Country' }, { name: 'State'}],
        columns: [{ name: 'Product', showNoDataItems: true }, { name: 'Date'
}],
        values: [{ name: 'Amount' }, { name: 'Quantity' }],
        showHeaderWhenEmpty: false
      },
      height: 350,
      showFieldList: true
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/pivotNullData-cs1" %}

Customizing loading indicator in Vue Pivotview component

You can customize the appearance of the loading indicator in the pivot table by using the spinnerTemplate property. This property accepts an HTML string which can be used for appearance customization.

You can also disable the loading indicator by setting spinnerTemplate to empty string.

APP.VUE

```
<template>
  <div id="app">
    <ejs-pivotview id="pivotview" :dataSourceSettings="dataSourceSettings"
:spinnerTemplate="spinnerTemplate" :height="height"> </ejs-pivotview>
  </div>
</template>
<script>
  import Vue from "vue";
  import { PivotViewPlugin, IDataSet } from "@syncfusion/ej2-vue-pivotview";
  import { DataManager, WebApiAdaptor, Query, ReturnOption } from
'@syncfusion/ej2-data';
  Vue.use(PivotViewPlugin);
  let remoteData: DataManager = new DataManager({
    url: "https://bi.syncfusion.com/northwindservice/api/orders",
    adaptor: new WebApiAdaptor(),
    crossDomain: true
```

```
  });
  export default {
    data () {
      return {
        dataSourceSettings: {
          dataSource: remoteData as DataManager,
          expandAll: true,
          filters: [],
          columns: [{ name: 'ProductName', caption: 'Product Name' }],
          rows: [{ name: 'ShipCountry', caption: 'Ship Country' }, { name:
'ShipCity', caption: 'Ship City' }],
          formatSettings: [{ name: 'UnitPrice', format: 'C0' }],
          values: [{ name: 'Quantity' }, { name: 'UnitPrice', caption: 'Unit
Price' }]
        },
        spinnerTemplate: '<i class="fa fa-cog fa-spin fa-3x fa-fw"></i>',
        height: 350
      }
    }
  }
</script>
<style>
  @import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/common-cs11" %}

Changing the pivotview component minimum height in Vue Pivotview component

The minHeight property allows you to change the minimum height for the pivot table control. For the pivot table control, the default minimum height is **300px**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList" :dataBound="ondataBound">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 200,
      showFieldList: true
    }
  },
  methods: {
    ondataBound: function() {
     let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
     pivotGridObj.minHeight = 200;
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs132" %}

Chart based on pivot table selection in Vue Pivotview component

The cell selection support is enabled using the allowSelection property and its type and mode are configured using the selectionSettings property. The cellSelected event gets fired on every selection operation performed in the pivot table. This event returns the selected cell informations, like row header name, column header name, measure name, and value. Based on this information, the chart control will be plotted.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview_chart" ref="pivotview_chart"
:dataSourceSettings="dataSourceSettings"
            :gridSettings="gridSettings" :dataBound="dataBound"
:cellSelected="cellSelected" :width="width"
            :height="height">
        </ejs-pivotview>
        <br />
        <ejs-chart id="chart" ref="chart" :title="title"
:primaryXAxis="primaryXAxis" :primaryYAxis="primaryYAxis"
            :legendSettings="legendSettings" :tooltip="tooltip">
            <e-series-collection>
                <e-series> </e-series>
            </e-series-collection>
        </ejs-chart>
    </div>
</template>
<script>
```

```
import Vue from "vue";
import {
    PivotViewPlugin,
    PivotCellSelectedEventArgs,
    FieldList,
} from "@syncfusion/ej2-vue-pivotview";
import { GridSettings } from "@syncfusion/ej2-
pivotview/src/pivotview/model/gridsettings";
import {
    ChartPlugin,
    ColumnSeries,
    Category,
    Legend,
    Tooltip,
} from "@syncfusion/ej2-vue-charts";
import { renewableEnergy } from "./datasource.js";
Vue.use(PivotViewPlugin);
Vue.use(ChartPlugin);
let onInit: boolean = true;
let measureList: { [key: string]: string } = {};
let chart: Chart;
let selectedCells: CellSelectedObject[];
let chartSeries: SeriesModel[];
export default {
    data() {
        return {
            dataSourceSettings: {
                dataSource: renewableEnergy,
                expandAll: true,
                enableSorting: true,
                drilledMembers: [
                    { name: "Year", items: ["FY 2015", "FY 2017", "FY 2018"]
},
                ],
                formatSettings: [{ name: "ProCost", format: "C0" }],
                rows: [{ name: "Year", caption: "Production Year" }],
                columns: [
                    { name: "EnerType", caption: "Energy Type" },
                    { name: "EneSource", caption: "Energy Source" },
                ],
                values: [{ name: "ProCost", caption: "Revenue Growth" }],
                filters: [],
            },
            gridSettings: {
                columnWidth: 120,
                allowSelection: true,
                selectionSettings: {
                    mode: "Cell",
                    type: "Multiple",
                    cellSelectionMode: "Box",
                },
            },
            width: "100%",
            height: 290,
            primaryXAxis: {
                valueType: "Category",
            },
```

```
            primaryYAxis: {
                labelFormat: "${value}K",
            },
            legendSettings: {
                visible: true,
            },
            title: "Sales Analysis",
            tooltip: {
                enable: true,
            },
        };
    },
    provide: {
        chart: [ColumnSeries, Category, Legend, Tooltip],
    },
    methods: {
        frameChartSeries: function () {
            let pivotGridObj =
                document.getElementById("pivotview_chart").ej2_instances[0];
            let columnGroupObject: { [key: string]: { x: string, y: number
}[] } = {};
            let valuesContent = pivotGridObj.engineModule.valueContent;
            if (onInit) {
                for (let cCnt = 0; cCnt < valuesContent.length; cCnt++) {
                    for (let cellIndex of Object.keys(valuesContent[cCnt]))
{
                        let cell = valuesContent[cCnt][Number(cellIndex)];
                        if (cell.columnHeaders && cell.rowHeaders) {
                            let columnSeries =

pivotGridObj.dataSourceSettings.values.length > 1
                                    ? cell.columnHeaders.toString() +
                                    " ~ " +
                                    measureList[cell.actualText]
                                    : cell.columnHeaders.toString();
                            if (columnGroupObject[columnSeries]) {
                                columnGroupObject[columnSeries].push({
                                    x: cell.rowHeaders.toString(),
                                    y: Number(cell.value),
                                });
                            } else {
                                columnGroupObject[columnSeries] = [
                                    { x: cell.rowHeaders.toString(), y:
Number(cell.value) },
                                ];
                            }
                        }
                    }
                }
            } else {
                for (let cell of selectedCells) {
                    if (cell.measure !== "") {
                        let columnSeries =
                            pivotGridObj.dataSourceSettings.values.length >
1 &&
                                measureList[cell.measure]
                                ? cell.columnHeaders.toString() +
```

```
                                    " ~ " +
                                    measureList[cell.measure]
                                    : cell.columnHeaders.toString();
                            if (columnGroupObject[columnSeries]) {
                                columnGroupObject[columnSeries].push({
                                    x:
                                        cell.rowHeaders == ""
                                            ? "Grand Total"
                                            : cell.rowHeaders.toString(),
                                    y: Number(cell.value),
                                });
                            } else {
                                columnGroupObject[columnSeries] = [
                                    {
                                        x:
                                            cell.rowHeaders == ""
                                                ? "Grand Total"
                                                : cell.rowHeaders.toString(),
                                        y: Number(cell.value),
                                    },
                                ];
                            }
                        }
                    }
                }
                let columnKeys: string[] = Object.keys(columnGroupObject);
                let chartSeries: SeriesModel[] = [];
                for (let key of columnKeys) {
                    chartSeries.push({
                        dataSource: columnGroupObject[key],
                        xName: "x",
                        yName: "y",
                        type: "Column",
                        name: key,
                    });
                }
                return chartSeries;
            },
            chartUpdate: function () {
                let pivotGridObj =
                    document.getElementById("pivotview_chart").ej2_instances[0];
                let chart = document.getElementById("chart").ej2_instances[0];
                if (onInit) {
                    onInit = false;
                } else {
                    chart.series = chartSeries;
                    chart.primaryXAxis.title =
pivotGridObj.dataSourceSettings.rows
                        .map(function (args) {
                            return args.caption || args.name;
                        })
                        .join(" ~ ");
                    chart.primaryYAxis.title =
pivotGridObj.dataSourceSettings.values
                        .map(function (args) {
                            return args.caption || args.name;
                        })
```

```
                    .join(" ~ ");
                chart.refresh();
            }
        },
        dataBound: function () {
            let pivotGridObj =
                document.getElementById("pivotview_chart").ej2_instances[0];
            if (onInit) {
                for (let value of pivotGridObj.dataSourceSettings.values) {
                    measureList[value.name] = value.caption || value.name;
                }
                pivotGridObj.grid.selectionModule.selectCellsByRange(
                    { cellIndex: 1, rowIndex: 1 },
                    { cellIndex: 3, rowIndex: 3 }
                );
                chartSeries = this.frameChartSeries();
                this.chartUpdate();
            }
        },
        cellSelected: function (args: PivotCellSelectedEventArgs) {
            selectedCells = args.selectedCellsInfo;
            if (selectedCells && selectedCells.length > 0) {
                chartSeries = this.frameChartSeries();
                this.chartUpdate();
            }
        },
    },
};
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/renewableEnergy-cs1" %}

Drill through grid cell edit type in Vue Pivotview component

Using the drillThrough event in the pivot table, you can define the edit type of a particular column in the grid present inside the drill-through dialog. To do so, check the column name in the drillThrough event and then specify the edit type of that column using the gridColumns.editType event argument.

The gridColumns.editType property must be set based on the column's data type. For example, the string data type will not be applicable for the numeric text box edit type.

- NumericTextBox control for integer, double, and decimal data types.
- TextBox control for string data type.
- DropDownList control to show all unique values related to that field.
- CheckBox control for boolean data type.
- DatePicker control for date data type.
- DateTimePicker control for date time data type.

In the below example, the data type of the `Country` column is set to `DropDownList`.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :editSettings="editSettings"> </ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, DrillThrough, DrillThroughEventArgs } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      editSettings: { allowAdding: true, allowDeleting: true, allowEditing:
true, mode: 'Normal' }
    }
  },
  provide: {
        pivotview: [DrillThrough]
  },
  methods: {
    drillThrough:function(args: DrillThroughEventArgs) {
        for (var i = 0; i < args.gridColumns.length; i++) {
            if (args.gridColumns[i].field === 'Country') {
                args.gridColumns[i].editType = 'dropdownedit';
                //args.gridColumns[i].editType = 'numericedit';
                //args.gridColumns[i].editType = 'textedit';
                //args.gridColumns[i].editType = 'booleanedit';
                //args.gridColumns[i].editType = 'datepickeredit';
                //args.gridColumns[i].editType = 'datetimepickeredit';
            }
        }
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs138" %}

Copyright © 2001 -2024 Syncfusion Inc.

Show field list when pivot table empty in Vue Pivotview component

When there are no fields in a pivot table's row, column, value, and filter axes, a field list can still be displayed. To do so, use the dataBound event and call the onShowFieldList method as shown below.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList" :dataBound="ondataBound">
</ejs-pivotview>
    </div>
</template>
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        allowLabelFilter: true,
        allowValueFilter: true,
        drilledMembers: [{ name: 'Country', items: ['France'] }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      showFieldList: true
    }
  },
  methods: {
    ondataBound: function() {
     let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
     pivotGridObj.pivotFieldListModule.dialogRenderer.fieldListDialog.target
= document.body;
     if (pivotGridObj && pivotGridObj.dataSourceSettings.values.length ===
0) {
        (pivotGridObj.pivotFieldListModule.dialogRenderer as
any).onShowFieldList();
     }
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
```

```
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs140" %}

Apply custom style to pivot cells in Vue Pivotview component

The queryCellInfo event in gridSettings can be used to apply custom style to row and value cells, and the headerCellInfo event in gridSettings can be used to apply custom styles to column cells.

In the following example, a custom style has been applied to the column header **"Sold Amount"** under **"FY 2016"** via the headerCellInfo event and to the row header **"Germany"** and its aggregated value via the queryCellInfo event by adding the **"e-custom"** class to the cell element.

APP.VUE

```
<template>
  <div id="app">
    <ejs-pivotview id="pivotview" :dataSourceSettings="dataSourceSettings"
:height="height" :gridSettings="gridSettings" >
    </ejs-pivotview>
</div>
</template >
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        rows: [{ name: 'Country' }, { name: 'Products' }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Amount', caption: 'Sold Amount' }, { name: 'Sold',
caption: 'Units Sold' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
      },
      height: 350,
      gridSettings: {
        queryCellInfo: function (args) {
            let colIndex = Number(args.cell.getAttribute('data-colindex'));
            let cells = args.data[colIndex] ? args.data[colIndex] : {};
            // Here by using 'actualText' option, a custom class can be
added to the specific row header and its value to apply custom style.
            if (cells.actualText === 'Germany') {
                args.cell.classList.add('e-custom');
            } else if (cells.actualText === 'Amount' &&
                cells.columnHeaders === 'FY 2016' && cells.rowHeaders ===
'Germany') {
                args.cell.classList.add('e-custom');
            }
        },
```

```
        headerCellInfo: function (args) {
            let customAttributes = args.cell.column.customAttributes;
            // Here custom class can be added to the specific column header
by using unique level name, to apply custom style.
            if (args.node.classList.contains('e-columnsheader') &&
customAttributes &&
                customAttributes.cell.valueSort.levelName === 'FY 2016.Sold
Amount') {
                args.node.classList.add('e-custom');
            }
        }
       },
    }
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs262" %}

Note: The **dot(.)** character in **FY 2016.Sold Amount** is used by default to identify the header levels in the pivot table's row and column. It can be changed by setting the headerDelimiter in the valueSortSettings property to any other delimiter instead of the default separator.

Show tooltip for row and column headers in Vue Pivotview component

You can create and display the tooltip for each row and column header(s) in the pivot table by using an external tooltip component via the dataBound event.

APP.VUE

```
<template>
  <div id="app">
    <ejs-pivotview id="pivotview" :dataSourceSettings="dataSourceSettings"
:height="height" :showFieldList="showFieldList"
        :dataBound="dataBound" > </ejs-pivotview>
</div>
</template >
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
let headerTooltip;
function beforeRender(args) {
    if (args.target.parentElement.querySelector('.e-rowsheader')) {
        // Here you can set custom content for row header(s) tooltip from
its cell information.
        var index = Number(args.target.getAttributeNode('index').value);
        var colIndex = Number(args.target.getAttributeNode('data-
colindex').value);
        var cell = pivotTableObj.engineModule.pivotValues[index][colIndex];
        var valueText = cell.valueSort ? cell.valueSort : '';
        if (cell.formattedText !== 'Grand Total') {
            this.content =
```

```
            '<div>' +
            'FieldName: ' +
            valueText.axis +
            '</br>' +
            'Text: ' +
            cell.formattedText +
            '</div>';
        } else {
            this.content =
            '<div>' +
            'FieldName: ' +
            valueText.uniqueName +
            '</br>' +
            'Text: ' +
            cell.formattedText +
            '</div>';
        }
    } else {
        // Here you can set custom content for column header(s) tooltip from
its cell information.
        if (args.target.querySelector('.e-cellvalue')) {
            this.content = args.target.querySelector('.e-
cellvalue').innerText;
        } else if (args.target.querySelector('.e-headertext')) {
            this.content = args.target.querySelector('.e-
headertext').innerText;
        } else if (args.target.querySelector('.e-stackedheadercelldiv')) {
            this.content = args.target.querySelector('.e-
stackedheadercelldiv').innerText;
        } else {
            this.content = '';
        }
    }
}
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        rows: [{ name: 'Country' }, { name: 'Products' }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{name:'Amount', caption:'Sold Amount'}, {name:'Sold',
caption:'Units Sold'}],
        formatSettings: [{name:'Amount', format:'C0'}],
      },
      height: 350,
      showFieldList: true
    }
  },
  methods: {
    dataBound: function () {
        let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
        if (!headerTooltip) {
            headerTooltip = new ej.popups.Tooltip({
```

```
                target: 'td.e-rowsheader,th.e-columnsheader', beforeRender:
beforeRender
            });
            headerTooltip.appendTo(pivotGridObj.element);
        }
    },
  }
  provide: {
    pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs263" %}

Hide specific columns in Vue Pivotview component

By using the columnRender event in the gridSettings, you can hide specific column(s) in the pivot table. In the example below, the **"Units Sold"** column under **"FY 2016"** is hidden by setting its **visible** property to **false** via the columnRender event.

Note: The **dot(.)** character in **FY 2016.Units Sold** is used by default to identify the header levels in the pivot table's row and column. It can be changed by setting the headerCellInfo in the queryCellInfo property to any other delimiter instead of the default separator.

APP.VUE

```
<template>
  <div id="app">
    <ejs-pivotview id="pivotview" :dataSourceSettings="dataSourceSettings"
:height="height"
        :gridSettings="gridSettings" ></ejs-pivotview>
</div>
</template >
<script>
import Vue from "vue";
import { PivotViewPlugin } from "@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        rows: [{ name: 'Country' }, { name: 'Products' }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{name:'Amount', caption:'Sold Amount'}, {name:'Sold',
caption:'Units Sold'}],
        formatSettings: [{name:'Amount', format:'C0'}],
      },
      height: 350,
      gridSettings: {
```

```
        columnRender: function(args){
            for (var i = 1; i < args.columns.length; i++) {
                if (args.stackedColumns[i].customAttributes &&

args.stackedColumns[i].customAttributes.cell.valueSort.levelName === 'FY
2016.Units Sold') {
                    args.stackedColumns[i].visible = false;
                }
            }
        }
      },
    }
  },
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs264" %}

<!-- markdownlint-disable MD009 -->

Export table and chart into the same document using toolbar in vue Pivotview component

Even if the displayOption.view property is set to **Both** in the pivot table, you can only export either the table or the chart to the PDF document based on the current value set in the displayOption.primary property. But, to export both the table and the chart to the same PDF document, use the pdfExport method during the actionBegin event invoke.

In the following example, the built-in export action can be restricted by setting the args.cancel option to **true** in the actionBegin event, and both the table and the chart can be exported by calling the pdfExport method and setting the exportBothTableAndChart argument to **true**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-pivotview id="pivotview"
:dataSourceSettings="dataSourceSettings" :height="height"
        :showToolbar="showToolbar" :allowPdfExport="allowPdfExport"
:displayOption="displayOption"
        :showFieldList="showFieldList" :toolbar="toolbar"
:actionBegin="actionBegin"
        :enableVirtualization="enableVirtualization"> </ejs-pivotview>
</div>
</template >
<script>
import Vue from "vue";
import { PivotViewPlugin, Toolbar, FieldList, PDFExport } from
"@syncfusion/ej2-vue-pivotview";
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
export default {
  data () {
    return {
      dataSourceSettings: {
```

```
        dataSource: pivotData,
        expandAll: false,
        rows: [{ name: 'Country' }, { name: 'Products' }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Amount', caption: 'Sold Amount' }, { name: 'Sold',
caption: 'Units Sold' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
      },
      height: 350,
      displayOption: { view:'Both' },
      toolbar: ['Grid', 'Chart', 'Export', 'FieldList'],
      showFieldList: true,
      showToolbar: true,
      allowPdfExport: true,
      enableVirtualization: true
    }
  },
  methods: {
    actionBegin: function (args) {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      if (args.actionName == 'PDF export') {
          args.cancel = true;
          pivotGridObj.pdfExport({}, false, null, false, true);
      }
    }
  },
  provide: {
        pivotview: [Toolbar, FieldList, PDFExport]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs265" %}

<!-- markdownlint-disable MD009 -->

Add custom aggregation type to the menu in Vue Pivotview component

By using the dataBound event, you can add your own custom aggregate type(s) to the pivot table's aggregate menu.

In the following example, we have added the aggregation types **CustomAggregateType 1** and **CustomAggregateType 2** to the aggregate menu. The calculation for those aggregated types can be done using the aggregateCellInfo event.

APP.VUE

```
<template>
  <div id="app">
    <ejs-pivotview id="pivotview" :dataSourceSettings="dataSourceSettings"
:height="height" :dataBound="ondataBound"
```

```
        :aggregateCellInfo="aggregateCell" :showFieldList="showFieldList" >
</ejs-pivotview>
</div>
</template >
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList} from "@syncfusion/ej2-vue-pivotview";
import { L10n } from '@syncfusion/ej2-base';
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
L10n.load({
     'en-US': {
          pivotview: {
               CustomAggregateType1: 'Custom Aggregate Type 1',
               CustomAggregateType2: 'Custom Aggregate Type 2',
          },
          pivotfieldlist: {
               CustomAggregateType1: 'Custom Aggregate Type 1',
               CustomAggregateType2: 'Custom Aggregate Type 2',
          }
     }
});
let SummaryType = [
    'Sum',
    'Count',
    'DistinctCount',
    'Avg',
    'CustomAggregateType1',
    'CustomAggregateType2'
];
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        rows: [{ name: 'Country' }, { name: 'Products' }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Amount', caption: 'Sold Amount' }, { name: 'Sold',
caption: 'Units Sold' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
      },
      height: 350,
      showFieldList: true
    }
  },
  methods: {
    ondataBound: function() {
      let pivotObj = document.getElementById('pivotview').ej2_instances[0];
      pivotObj.getAllSummaryType = function () {
          return SummaryType;
      };
      pivotObj.pivotFieldListModule.aggregateTypes = SummaryType;
      pivotObj.pivotFieldListModule.getAllSummaryType = function () {
          return SummaryType;
      };
```

```
    },
    aggregateCell:function(args) {
      if (args.aggregateType === 'CustomAggregateType1') {
          args.value = args.value * 100;
      }
      if (args.aggregateType === 'CustomAggregateType2') {
          args.value = args.value / 100;
      }
    }
  },
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs266" %}

<!-- markdownlint-disable MD009 -->

Convert complex JSON to flat JSON and assign it to the pivot table in Vue Pivotview component

By default, flat JSON can only bind to the pivot table. However, you can connect complex JSON to the pivot table by converting it to flat JSON via code-behind and binding it to the pivot table using the dataSource property in the load event.

In the following example, the **complexToFlatJson()** method is used to convert complex JSON to flat JSON and bind it to the pivot table using the dataSource property, then modifying the field names in the rows and columns based on the converted flat JSON under dataSourceSettings in the load event.

APP.VUE

```
<template>
     <div id="app">
          <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings"
        :load="load" :height="height" :showFieldList="showFieldList" >
</ejs-pivotview>
</div>
</template >
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
Vue.use(PivotViewPlugin);
var parentProp = {};
var dataSource;
function complexToFlatJson(data) {
    var flatArray = [];
    var flatObject = {};
    for (var index = 0; index < data.length; index++) {
        for (var prop in data[index]) {
            var value = data[index][prop];
            if (Array.isArray(value)) {
                for (var i = 0; i < value.length; i++) {
```

```
                    var childProp = [];
                    for (var inProp in value[i]) {
                        flatObject[inProp] = value[i][inProp];
                        var object = {
                            name: inProp,
                        };
                        childProp.push(object);
                    }
                    parentProp[prop] = childProp;
                }
            }
            else {
                flatObject[prop] = value;
            }
        }
        flatArray.push(flatObject);
        flatObject = {};
    }
    return flatArray;
}
function pivotData() {
    return [
        {
            CustomerID: 'VINET',
            Freight: 32.38,
            OrderDetails: [
                {
                    OrderID: 10248,
                    OrderDate: '1996-07-04T10:10:00.000Z',
                }
            ],
            ShipDetails: [
                {
                    ShipName: 'Vins et alcools Chevalier',
                    ShipAddress: "59 rue de l'Abbaye",
                    ShipCity: 'Reims',
                    ShipRegion: null,
                    ShipCountry: 'France',
                    ShippedDate: '1996-07-16T12:20:00.000Z',
                }
            ]
        },
        {
            CustomerID: 'GALED',
            Freight: 10.14,
            OrderDetails: [
                {
                    OrderID: 10366,
                    OrderDate: '1996-11-28T00:00:00.000Z',
                }
            ],
            ShipDetails: [
                {
                    ShippedDate: '1996-12-30T00:00:00.000Z',
                    ShipName: 'Galería del gastronómo',
                    ShipAddress: 'Rambla de Cataluña, 23',
                    ShipCity: 'Barcelona',
```

```
                    ShipRegion: null,
                    ShipCountry: 'Spain',
                }
            ]
        },
        {
            CustomerID: 'VAFFE',
            Freight: 13.55,
            OrderDetails: [
                {
                    OrderID: 10367,
                    OrderDate: '1996-12-02T00:00:00.000Z',
                }
            ],
            ShipDetails: [
                {
                    ShippedDate: '1996-12-30T00:00:00.000Z',
                    ShipName: 'Vaffeljernet',
                    ShipAddress: 'Smagsloget 45',
                    ShipCity: 'Århus',
                    ShipRegion: null,
                    ShipCountry: 'Denmark',
                }
            ]
        },
        {
            CustomerID: 'ERNSH',
            Freight: 101.95,
            OrderDetails: [
                {
                    OrderID: 10368,
                    OrderDate: '1996-11-29T00:00:00.000Z',
                }
            ],
            ShipDetails: [
                {
                    ShippedDate: '1996-12-30T00:00:00.000Z',
                    ShipName: 'Ernst Handel',
                    ShipAddress: 'Kirchgasse 6',
                    ShipCity: 'Graz',
                    ShipRegion: null,
                    ShipCountry: 'Austria',
                }
            ]
        },
        {
            CustomerID: 'SPLIR',
            Freight: 195.68,
            OrderDetails: [
                {
                    OrderID: 10369,
                    OrderDate: '1996-11-28T00:00:00.000Z',
                }
            ],
            ShipDetails: [
                {
                    ShippedDate: '1996-12-30T00:00:00.000Z',
```

```
                    ShipName: 'Split Rail Beer & Ale',
                    ShipAddress: 'P.O. Box 555',
                    ShipCity: 'Lander',
                    ShipRegion: 'WY',
                    ShipCountry: 'USA',
                },
            ]
        }
    ];
}
export default {
  data () {
    return {
      dataSourceSettings: {
        expandAll: true,
        enableSorting: true,
        dataSource: pivotData(),
        columns: [{ name: 'OrderDetails' }],
        values: [{ name: 'Freight', caption: 'Units Sold' }],
        rows: [{ name: 'ShipDetails' }],
        valueSortSettings: { headerDelimiter: ' - ' },
        formatSettings: [{ name: 'Amount', format: 'C0' }]
      },
      height: 350,
      showFieldList: true
    }
  },
  methods: {
     load: function (args) {
          dataSource =
JSON.parse(JSON.stringify(args.dataSourceSettings.dataSource));
          args.dataSourceSettings.dataSource =
complexToFlatJson(dataSource);
          var rows = [];
          for (var i = 0; i < args.dataSourceSettings.rows.length; i++) {
              if (args.dataSourceSettings.rows[i].name in parentProp) {
                  rows =
rows.concat(parentProp[args.dataSourceSettings.rows[i].name]);
              }
              else {
                  rows.push(args.dataSourceSettings.rows[i]);
              }
          }
          args.dataSourceSettings.rows = rows;
          var columns = [];
          for (var i = 0; i < args.dataSourceSettings.columns.length; i++) {
              if (args.dataSourceSettings.columns[i].name in parentProp) {
                  columns =
columns.concat(parentProp[args.dataSourceSettings.columns[i].name]);
              }
              else {
                  columns.push(args.dataSourceSettings.columns[i]);
              }
          }
          args.dataSourceSettings.columns = columns;
      },
  },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  provide: {
        pivotview: [FieldList]
    }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs268" %}

<!-- markdownlint-disable MD009 -->

Load desired report from the report list as default in vue Pivotview component

By default, the pivot table is displayed with the report bound at the code-behind. To load a desired report from the previously saved report collection during initial rendering, set the desired report name in the dataBound event, along with the additional report-based customization code shown below.

APP.VUE

```
<template>
  <div id="app">
    <ejs-pivotview id="pivotview" ref="pivotview"
:dataSourceSettings="dataSourceSettings" :gridSettings="gridSettings"
:height="height"
    :allowExcelExport="allowExcelExport"
:allowConditionalFormatting="allowConditionalFormatting"
:allowPdfExport="allowPdfExport"
    :showToolbar="showToolbar"
:allowNumberFormatting="allowNumberFormatting"
:allowCalculatedField="allowCalculatedField" :showFieldList="showFieldList"
    :toolbar="toolbar" :saveReport="saveReport" :loadReport="loadReport"
:fetchReport="fetchReport" :renameReport="renameReport"
:removeReport="removeReport"
    :newReport="newReport" :displayOption="displayOption"
:dataBound="ondataBound" :load="load"> </ejs-pivotview>
</div>
</template >
<script>
import Vue from "vue";
import {
  PivotViewPlugin,
  FieldList,
  CalculatedField,
  Toolbar,
  PDFExport,
  ExcelExport,
  ConditionalFormatting,
  NumberFormatting
} from "@syncfusion/ej2-vue-pivotview";
import{ getInstance, select} from '@syncfusion/ej2-base';
import {DropDownList} from '@syncfusion/ej2-dropdowns';
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
function loadReport(args) {
    var reportCollection = [];
```

```
    if (localStorage.pivotviewReports && localStorage.pivotviewReports !==
"") {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
    }
    reportCollection.map(function (item) {
        if (args.reportName === item.reportName) {
            args.report = item.report;
        }
    });
    if (args.report) {
        let pivotObj =
document.getElementById('pivotview').ej2_instances[0];
        pivotObj.dataSourceSettings =
JSON.parse(args.report).dataSourceSettings;
    }
}
var isInitial = true;
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Sold', caption: 'Units Sold' }, { name: 'Amount',
caption: 'Sold Amount' }],
        rows: [{ name: 'Country' }, { name: 'Products' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
        filters: []
      },
      height: 350,
      gridSettings: { columnWidth: 140 },
      allowExcelExport: true,
      allowConditionalFormatting: true,
      allowNumberFormatting: true,
      allowPdfExport: true,
      displayOption: { view:'Both' },
      showToolbar: true,
      allowCalculatedField: true,
      showFieldList: true,
      toolbar: [
        "New",
        "Save",
        "SaveAs",
        "Rename",
        "Remove",
        "Load",
        "Grid",
        "Chart",
        "Export",
        "SubTotal",
        "GrandTotal",
        "ConditionalFormatting",
        "NumberFormatting",
        "FieldList"
      ]
```

```
    };
  },
  methods: {
    saveReport: function(args) {
      let reports = [];
      let isSaved = false;
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reports = JSON.parse(localStorage.pivotviewReports);
      }
      if (args.report && args.reportName && args.reportName !== "") {
        reports.map(function(item) {
          if (args.reportName === item.reportName) {
            item.report = args.report;
            isSaved = true;
          }
        });
        if (!isSaved) {
          reports.push(args);
        }
        localStorage.pivotviewReports = JSON.stringify(reports);
      }
    },
    fetchReport: function(args) {
      let reportCollection = [];
      let reeportList = [];
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
      }
      reportCollection.map(function(item) {
        reeportList.push(item.reportName);
      });
      args.reportName = reeportList;
    },
    removeReport: function(args) {
      let reportCollection = [];
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
      }
      for (let i = 0; i < reportCollection.length; i++) {
        if (reportCollection[i].reportName === args.reportName) {
          reportCollection.splice(i, 1);
        }
      }
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        localStorage.pivotviewReports = JSON.stringify(reportCollection);
```

```
      }
    },
    renameReport: function(args) {
      let reportCollection = [];
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        reportCollection = JSON.parse(localStorage.pivotviewReports);
      }
      reportCollection.map(function(item) {
        if (args.reportName === item.reportName) {
          item.reportName = args.rename;
        }
      });
      if (
        localStorage.pivotviewReports &&
        localStorage.pivotviewReports !== ""
      ) {
        localStorage.pivotviewReports = JSON.stringify(reportCollection);
      }
    },
    newReport: function() {
      let pivotGridObj =
document.getElementById('pivotview').ej2_instances[0];
      pivotGridObj.setProperties(
        {
          dataSourceSettings: {
            columns: [],
            rows: [],
            values: [],
            filters: []
          }
        },
        false
      );
    },
    loadReport: loadReport,
    ondataBound: function () {
        let pivotObj =
document.getElementById('pivotview').ej2_instances[0];
        if (pivotObj && isInitial) {
            isInitial = false;
            pivotObj.toolbarModule.action = 'Load';
            let reportList = getInstance(select('#' + pivotObj.element.id +
'_reportlist', pivotObj.element), DropDownList);
            reportList.value = 'Default report';
            loadReport({ reportName: 'Default report' });
        }
    },
    load: function () {
      var dataSourceSettings = {
        dataSource: pivotData,
        columns: [{ name: 'Year' }],
        enableSorting: true,
        allowLabelFilter: true,
        values: [{ name: 'Sold', caption: 'Units Sold' }],
```

```
        allowValueFilter: true,
        formatSettings: [{ name: 'Sold', format: 'C0' }],
        rows: [{ name: 'Country' }],
      };
      var displayOption = { view: 'Both' };
      var gridSettings = {columnWidth: 100};
      var report = { dataSourceSettings: dataSourceSettings, displayOption:
displayOption, gridSettings: gridSettings };
      var reports = [
        {
          report: JSON.stringify(report),
          reportName: 'Default report',
        },
      ];
      localStorage.pivotviewReports = JSON.stringify(reports);
    },
  },
  provide: {
    pivotview: [
      FieldList,
      CalculatedField,
      Toolbar,
      PDFExport,
      ExcelExport,
      ConditionalFormatting,
      NumberFormatting
    ]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs269" %}

<!-- markdownlint-disable MD009 -->

Display string value to pivot table values in vue Pivotview component

End user can display string value to the pivot table's value cell by using the aggregateCellInfo event.

In the following example, each cell value of the **Sold** field's actual value has been assigned from its combination data sets obtained from the args.cellSets in the aggregateCellInfo event.

APP.VUE

```
<template>
  <div id="app">
    <ejs-pivotview id="pivotview" :dataSourceSettings="dataSourceSettings"
:height="height"
        :aggregateCellInfo="aggregateCell" :showFieldList="showFieldList" >
</ejs-pivotview>
</div>
</template >
<script>
import Vue from "vue";
import { PivotViewPlugin, FieldList } from "@syncfusion/ej2-vue-pivotview";
```

```
import { pivotData } from './pivotData.js';
Vue.use(PivotViewPlugin);
function secondsToHms(d) {
     d = Number(d);
     var h = Math.floor(d / 3600);
     var m = Math.floor((d % 3600) / 60);
     var s = Math.floor((d % 3600) % 60);
     return (
          ('0' + h).slice(-2) + ':' + ('0' + m).slice(-2) + ':' + ('0' +
s).slice(-2)
     );
}
export default {
  data () {
    return {
      dataSourceSettings: {
        dataSource: pivotData,
        expandAll: false,
        rows: [{ name: 'Country' }, { name: 'Products' }],
        columns: [{ name: 'Year', caption: 'Production Year' }, { name:
'Quarter' }],
        values: [{ name: 'Amount', caption: 'Sold Amount' }, { name: 'Sold',
caption: 'Units Sold' }],
        formatSettings: [{ name: 'Amount', format: 'C0' }],
      },
      height: 350,
      showFieldList: true
    }
  },
  methods: {
    aggregateCell:function(args) {
      if (args.fieldName === 'Sold') {
          args.value = secondsToHms(args.value);
    }
  }
  },
  provide: {
    pivotview: [FieldList]
  }
}
</script>
<style>
@import "@syncfusion/ej2-vue-pivotview/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/pivot-grid/default-cs267" %}

Predefined Dialogs

Getting Started with the Vue Predefined dialogs Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Predefined dialogs component

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The list of dependencies required to use the predefined dialogs in your application is given below:

```
|-- @syncfusion/ej2-vue-popups
|-- @syncfusion/ej2-vue-base
|-- @syncfusion/ej2-vue-buttons
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-buttons
```

Setting up the Vue 2 project

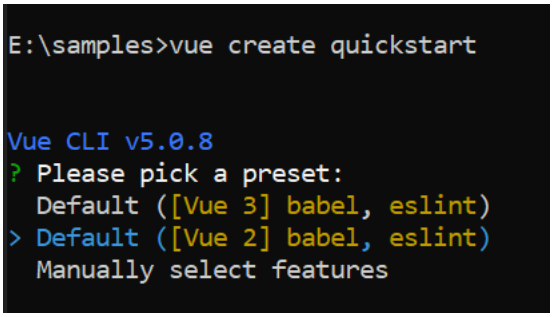
To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

```bash
npm install -g @vue/cli
vue create quickstart
cd quickstart
npm run serve
```

or

```bash
yarn global add @vue/cli
vue create quickstart
cd quickstart
yarn run serve
```

When creating a new project, choose the option `Default ([Vue 2] babel, eslint)` from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Predefined dialogs component as an example. Install the @syncfusion/ej2-vue-popups package by running the following command:

`bash

npm install @syncfusion/ej2-vue-popups --save

`

or

`bash

yarn add @syncfusion/ej2-vue-popups

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the predefined dialogs component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
</style>
```

The Custom Resource Generator (CRG) is an online web tool, which can be used to generate the custom script and styles for a set of specific components.

This web tool is useful to combine the required component scripts and styles in a single file.

Add Syncfusion Vue component

Follow the below steps to add the Vue Predefined dialogs component:

1\. First, import and register the Predefined dialogs component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonComponent } from "@syncfusion/ej2-vue-buttons";
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
components: {
'ejs-button': ButtonComponent
}
}
</script>
```

2\. In the template section, define the Predefined dialogs component.

~/SRC/APP.VUE

```
<template>
<div>
<div class="predefinedDialogs">
<ejs-button id="alertDlgBtn" v-on:click.native="alertBtnClick" cssClass="e-danger">Alert</ejs-button>
<span id="statusText"></span>
</div>
</div>
</template>
```

Creating Vue sample

Add the following code to the App.vue file in src directory.

`

<template>

<ejs-button id="alertDlgBtn" v-on:click.native="alertBtnClick" cssClass="e-danger">Alert</ejs-button>

</template>

<style>

@import "../node_modules/@syncfusion/ej2-base/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";

statusText {

font-size: 16px;

margin-top: 20px;

}

.predefinedDialogs {

height: 100%;

Copyright © 2001 -2024 Syncfusion Inc.

```
min-height: 350px;
}
</style>
<script>
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonComponent } from "@syncfusion/ej2-vue-buttons";
export default {
components: {
'ejs-button': ButtonComponent
},
data: function () {
return {};
},
methods: {
alertBtnClick: function () {
document.getElementById("statusText").style.display = "none";
DialogUtility.alert({
title: "Low battery",
content: "10% of battery remaining",
});
},
},
}
</script>
`
```

Show alert dialog

An alert dialog box used to display an errors, warnings, and information alerts that needs user awareness. This can be achieved by using the DialogUtility.alert method. The alert dialog is displayed along with the OK button. When user clicks on OK button, alert dialog will get closed.

In the below code example, alert dialog displayed on button click action.

Now run the npm run dev command in the console, it will build your application and open in the browser.

~/SRC/APP.VUE

```
<template>
  <div class="predefinedDialogs">
```

```
    <ejs-button id="alertDlgBtn" v-on:click.native="alertBtnClick"
cssClass="e-danger">Alert</ejs-button>
    <span id="statusText"></span>
  </div>
</template>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#statusText {
  font-size: 16px;
  margin-top: 20px;
}
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonComponent } from "@syncfusion/ej2-vue-buttons";
let dialogObj = undefined;
export default {
  components: {
    'ejs-button': ButtonComponent
  },
  data: function () {
    return {};
  },
  methods: {
    alertBtnClick: function () {
      document.getElementById("statusText").style.display = "none";
      dialogObj = DialogUtility.alert({
        title: "Low battery",
        content: "10% of battery remaining",
        okButton: { click: this.alertOkAction },
        position: { X: "center", Y: "center" },
      });
    },
    alertOkAction: function () {
      dialogObj.hide();
      document.getElementById("statusText").innerHTML =
        "The user closed the Alert dialog.";
      document.getElementById("statusText").style.display = "block";
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/alert-dialog-cs1" %}

Show confirm dialog

A confirm dialog box used to displays a specified message along with the OK and Cancel buttons. This can be achieved by using the DialogUtility.confirm method. It is used to get approval from the user,

Copyright © 2001 -2024 Syncfusion Inc.

and it appears before any critical action. After get approval from the user the dialog will disappear automatically.

In the below code example, the confirm dialog displayed on OK and Cancel button click action.

~/SRC/APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="confirmDlgBtn"  v-on:click.native="confirmBtnClick"
cssClass = "e-success">Confirm</ejs-button>
    <span id="statusText"></span>
  </div>
</template>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#statusText {
  font-size: 16px;
  margin-top: 20px;
}
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonComponent } from "@syncfusion/ej2-vue-buttons";
let dialogObj = undefined;
export default {
  components: {
    'ejs-button': ButtonComponent
  },
  data: function () {
    return {};
  },
  methods: {
    confirmBtnClick: function () {
        document.getElementById("statusText").style.display="none";
        dialogObj = DialogUtility.confirm({
            title: " Delete multiple items",
            content: "Are you sure you want to permanently delete these
items?",
            okButton: { click:this.confirmOkAction},
            cancelButton: { click:this.confirmCancelAction},
        });
      },
      confirmOkAction:function () {
          dialogObj.hide();
          document.getElementById("statusText").innerHTML="The user
confirmed the dialog box";
          document.getElementById("statusText").style.display="block";
      },
      confirmCancelAction:function(){
          dialogObj.hide();
```

```
            document.getElementById("statusText").innerHTML="The user canceled
the dialog box.";
            document.getElementById("statusText").style.display="block";
        }
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/confirm-dialog-cs1" %}

Show Prompt dialog

A prompt dialog is used to get the input from the user. When the user clicks the OK button the input value from the dialog is returned. If the user clicks the Cancel button the null value is returned. After getting the input from the user the dialog will disappear automatically.

In the below code example, the confirm dialog displayed on OK and Cancel button click action.

~/SRC/APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="promptDlgBtn"  v-on:click.native="promptBtnClick"
:isPrimary="true">Prompt</ejs-button>
    <span id="statusText"></span>
  </div>
</template>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#statusText {
  font-size: 16px;
  margin-top: 20px;
}
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonComponent } from "@syncfusion/ej2-vue-buttons";
let dialogObj = undefined;
let value ;
export default {
  components: {
    'ejs-button': ButtonComponent
  },
  data: function () {
    return {};
  },
  methods: {
     promptBtnClick: function () {
            document.getElementById("statusText").style.display="none";
            dialogObj = DialogUtility.confirm({
                title: "Join chat group",
```

```
                content:'<P>Enter your name:</p><input id= "inputEle"
type="text" name="Required" class="e-input" placeholder="Type here.." />',
                okButton: { click:this.promptOkAction },
                cancelButton: { click:this.promptCancelAction},
                position: { X: 'center', Y: 'center' }
            });
        },
    promptOkAction:function () {
        value = document.getElementById("inputEle").value;
        if(value ==""){
            dialogObj.hide();
            document.getElementById("statusText").innerHTML="The user's
input is returned as\" \"";
            document.getElementById("statusText").style.display="block";
        }
        else{
            dialogObj.hide();
            document.getElementById("statusText").innerHTML="The user's
input is returned as" +" "+ value;
            document.getElementById("statusText").style.display="block";
        }
        },
        promptCancelAction:function () {
            dialogObj.hide ();
            document.getElementById("statusText").innerHTML ="The user
canceled the prompt dialog";
            document.getElementById("statusText").style.display="block";
        }
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/prompt-dialog-cs1" %}

Getting Started with the Vue Predefined Dialogs Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Predefined Dialogs component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

Copyright © 2001 -2024 Syncfusion Inc.

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

To use the Vue Predefined Dialogs component in the project, the @syncfusion/ej2-vue-popups package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-popups --save

`

or

`bash

yarn add @syncfusion/ej2-vue-popups

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Predefined Dialogs component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

The order of importing CSS styles should be in line with its dependency graph.

Adding Syncfusion Vue3 Predefined Dialogs in the application

You have completed all the necessary configurations needed for rendering the Syncfusion Vue component. Now, you are going to add the Predefined Dialogs using following steps.

- Import the DialogUtility in the `

~/SRC/APP.VUE

```
<script>
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class="predefinedDialogs">
<button id="alertDlgBtn" @click="alertBtnClick" class="e-
danger">Alert</button>
</div>
</template>
<script>
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
export default {
setup() {
const alertBtnClick = () => {
DialogUtility.alert({
title: "Low battery",
content: "10% of battery remaining",
});
};
return {
alertBtnClick
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
.predefinedDialogs {
height: 100%;
min-height: 350px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="predefinedDialogs">
<button id="alertDlgBtn" @click="alertBtnClick" class="e-
danger">Alert</button>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
export default {
data() {
return {};
},
methods: {
alertBtnClick() {
DialogUtility.alert({
title: "Low battery",
content: "10% of battery remaining",
});
},
},
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
.predefinedDialogs {
height: 100%;
min-height: 350px;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

web server will be initiated, open the quick start app in the browser at port localhost:8080.

Draggable in Vue Predefined dialogs component

The predefined dialogs supports dragging within its target container by grabbing the dialog header, which allows the user to reposition the dialog dynamically by using isDraggable property.

Alert drag

APP.VUE

```
<template>
  <div class="predefinedDialogs">
      <ejs-button id="alertDlgBtn" v-on:click.native="alertBtnClick"
cssClass="e-danger">Alert</ejs-button>
  </div>
</template>
<style>
```

```
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
    alertBtnClick: function () {
      DialogUtility.alert({
        title: 'Low battery',
        width: '250px',
        isDraggable: true,
        content: '10% of battery remaining',
      });
    }
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/alert-drag-cs1" %}

Confirm drag

APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="confirmDlgBtn" v-on:click.native="confirmBtnClick"
cssClass = "e-success">Confirm</ejs-button>
  </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  data: function () {
    return {};
  },
  methods: {
    confirmBtnClick: function () {
      DialogUtility.confirm({
        title: 'Delete multiple items',
        content: "Are you sure you want to permanently delete these items?",
        width: '300px',
        isDraggable:true
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/confirm-drag-cs1" %}

Prompt drag

APP.VUE

```
<template>
  <div class="predefinedDialogs">
      <ejs-button id="promptDlgBtn" v-on:click.native="promptBtnClick"
:isPrimary="true">Prompt</ejs-button>
  </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
     promptBtnClick: function () {
       DialogUtility.confirm({
        title: 'Join chat group',
        width:'250px',
        isDraggable:true,
        content: '<p>Enter your name: </p><input id= "inputEle" type="text"
name="Required" class="e-input" placeholder="Type here.." />'
      });
    },
  },
```

```
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/prompt-drag-cs1" %}

Animation in Vue Predefined dialogs component

The predefined dialogs can be animated during the open and close actions. Also, user can customize animation's delay, duration and effect of animation by using the animationSettings property.

In the below sample, Zoom effect is enabled. So, The Dialog will open with ZoomIn and close with ZoomOut effects.

Alert animation

APP.VUE

```
<template>
    <div class="predefinedDialogs">
        <ejs-button id="alertDlgBtn"  v-on:click.native="alertBtnClick"
cssClass="e-danger">Alert</ejs-button>
    </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
let dialogObj = undefined;
export default {
  data: function () {
    return {};
  },
  methods: {
    alertBtnClick: function () {
      dialogObj = DialogUtility.alert({
          title: 'Low battery',
          width: '250px',
          animationSettings: { effect: 'Zoom' },
          content: '10% of battery remaining'
      });
    }
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/alert-animation-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Confirm animation

APP.VUE

```
<template>
    <div class="predefinedDialogs">
        <ejs-button id="confirmDlgBtn"  v-on:click.native="confirmBtnClick"
cssClass = "e-success">Confirm</ejs-button>
    </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
     confirmBtnClick: function () {
        DialogUtility.confirm({
          title: 'Delete multiple items',
          content: "Are you sure you want to permanently delete these
items?",
          width: '300px',
          animationSettings: { effect: 'Zoom' }
        });
    }
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/confirm-animation-cs1" %}

Prompt animation

APP.VUE

```
<template>
    <div class="predefinedDialogs">
        <ejs-button id="promptDlgBtn"  v-on:click.native="promptBtnClick"
:isPrimary="true">Prompt</ejs-button>
    </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
     promptBtnClick: function () {
       DialogUtility.confirm({
        title: 'Join chat group',
        width:'250px',
        animationSettings: { effect: 'Zoom' },
        content: '<p>Enter your name: </p><input id= "inputEle" type="text"
name="Required" class="e-input" placeholder="Type here.." />'
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/prompt-animation-cs1" %}

Position in Vue Predefined dialogs component

Customize the dialog position by using the position property. The position can be represented with specific X and Y values.

- The PositionDataModel.X can be configured with a left, center, right or offset value. By default, the value is set as center.
- The PositionDataModel.Y can be configured with a top, center, bottom or offset value. By default, the value is set as center.

Alert position

APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="alertDlgBtn" v-on:click.native="alertBtnClick"
cssClass="e-danger">Alert</ejs-button>
  </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
```

```
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
    alertBtnClick: function () {
      DialogUtility.alert({
        title: "Low battery",
        width: "250px",
        content: "10% of battery remaining",
        position: { X: "center", Y: "center" },
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/alert-position-cs1" %}

Confirm position

APP.VUE

```
<template>
    <div class="predefinedDialogs">
      <ejs-button id="confirmDlgBtn" v-on:click.native="confirmBtnClick"
cssClass="e-success">Confirm</ejs-button>
    </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
    confirmBtnClick: function () {
```

```
      DialogUtility.confirm({
        title: "Delete multiple items",
        content: "Are you sure you want to permanently delete these items?",
        width: "300px",
        position: { X: "center", Y: "center" },
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/confirm-position-cs1" %}

Prompt position

APP.VUE

```
<template>
  <div class="predefinedDialogs">
      <ejs-button id="promptDlgBtn"  v-on:click.native="promptBtnClick"
:isPrimary="true">Prompt</ejs-button>
  </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
     promptBtnClick: function () {
       DialogUtility.confirm({
        title: "Join chat group",
        width:"250px",
        content: '<p>Enter your name: </p><input id= "inputEle" type="text"
name="Required" class="e-input" placeholder="Type here.." />',
        position: { X: 'center', Y: 'center' }
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/prompt-position-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Dimension in Vue Predefined dialogs component

Customize the predefined dialogs dimensions using the height and width properties.

You can specify the dimension values in both pixels and percentage format to change the default dialog width and height values.

Alert dimension

APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="alertDlgBtn" v-on:click.native="alertBtnClick"
cssClass="e-danger">Alert</ejs-button>
  </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
    alertBtnClick: function () {
      DialogUtility.alert({
        title: 'Not enough space',
        width: '300px',
        height: '200px',
        content: 'Delete certain files to free up space to store more
items.'
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/alert-dimension-cs1" %}

Confirm dimension

APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="confirmDlgBtn" v-on:click.native="confirmBtnClick"
cssClass="e-success">Confirm</ejs-button>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
    confirmBtnClick: function () {
      DialogUtility.confirm({
        title: 'Delete multiple items',
        content: "Are you sure you want to permanently delete these items?",
        width: '300px',
        height: '200px'
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/confirm-dimension-cs1" %}

Prompt dimension

APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="promptDlgBtn"  v-on:click.native="promptBtnClick"
:isPrimary="true">Prompt</ejs-button>
  </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
     promptBtnClick: function () {
       DialogUtility.confirm({
        title: 'Join chat group',
        width: '300px',
        height: '250px',
        content: '<p>Enter your name: </p><input id= "inputEle" type="text"
name="Required" class="e-input" placeholder="Type here.." />'
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/prompt-dimension-cs1" %}

Max-width and max-height

To have a restricted max-width and max-height dialog dimension, you need to specify the max-width, max-height CSS properties for the component's container element by using the cssClass property. The max-height value is calculated in source level and set to the dialog. so, need to override the max-height property.

Use the following code to customize the max-width and max-height for alert dialog:

APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="alertDlgBtn" v-on:click.native="alertBtnClick"
cssClass="e-danger">Alert</ejs-button>
  </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
 .e-alert-dialog.customClass {
    max-width: 350px;
    max-height: 250px !important;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data: function () {
    return {};
  },
  methods: {
    alertBtnClick: function () {
      DialogUtility.alert({
       title: 'About syncfusion succinctly series',
        content: 'In the Succinctly series, Syncfusion created a robust,
free library of more than 130 technical e-books formatted for PDF, Kindle,
and EPUB.Each title in the Succinctly series is written by a carefully
chosen expert and provides essential content in about 100 pages. The
Succinctly series was born in 2012 out of a desire to provide concise
technical e-books for software developers.',
        cssClass : 'customClass'
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/maxWidth-maxHeight-cs1" %}

Min-width and min-height

To have a restricted min-width and min-height dialog dimension, you need to specify the min-width, min-height CSS properties for the component's container element by using the cssClass property.

Use the following code to customize the min-width and min-height for alert dialog:

APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="alertDlgBtn" v-on:click.native="alertBtnClick"
cssClass="e-danger">Alert</ejs-button>
  </div>
</template>
<style>
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
.e-dialog.customClass {
  min-width: 300px;
  min-height: 200px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  data: function () {
    return {};
  },
  methods: {
    alertBtnClick: function () {
      DialogUtility.alert({
        title: "About syncfusion succinctly series",
        content: "The succinctly series was born in 2012.",
        cssClass: "customClass",
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/minWidth-minHeight-cs1" %}

Customization in Vue Predefined dialogs component

You can customize the predefined dialogs buttons by using below properties.

- okButton - Use this property to customize **OK** button text.
- cancelButton - Use this property to customize **Cancel** button text.

Use the following code snippet for **alert**, **confirm** and **prompt** to customize the predefined dialogs action buttons.

For alert dialog , customized the default dialog button content as Dismiss by using the text property.

For confirm dialog, customized the default dialog buttons content as Yes and No by using the text property and also customized the dialog button icons by using icon property.

For prompt dialog , customized the default dialog buttons content as Connect and Close by using text property.

Alert action button

APP.VUE

```
<template>
    <div class="predefinedDialog">
      <ejs-button id="alertDlgBtn" v-on:click.native="alertBtnClick"
cssClass="e-danger">Alert</ejs-button>
    </div>
</template>
<style>
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialog {
  height: 100%;
  min-height: 350px;
}
.e-dialog.customClass {
  min-width: 300px;
  min-height: 200px;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
    alertBtnClick: function () {
      DialogUtility.alert({
        title: 'Good job!',
        content: 'You clicked the alert button!',
        width:'250px',
        okButton: { text: 'Done'}
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/alert-action-btn-cs1" %}

Confirm action buttons

APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="confirmDlgBtn" v-on:click.native="confirmBtnClick"
cssClass="e-success">Confirm</ejs-button>
  </div>
</template>
<style>
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
    confirmBtnClick: function () {
      DialogUtility.confirm({
        title: 'Delete file',
```

```
        content: "Are you sure you want to permanently delete these file?",
        width: '300px',
        okButton: { text: 'Yes', icon:'e-icons e-check'},
        cancelButton: {text:'No', icon:'e-icons e-close'}
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/confirm-action-btn-cs1" %}

Prompt action buttons

APP.VUE

```
<template>
  <div class="predefinedDialogs">
      <ejs-button id="promptDlgBtn"  v-on:click.native="promptBtnClick"
:isPrimary="true">Prompt</ejs-button>
  </div>
</template>
<style>
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
     promptBtnClick: function () {
       DialogUtility.confirm({
        title: 'Join chat group',
        width: '300px',
        content: '<p>Enter your name: </p><input id= "inputEle" type="text"
name="Required" class="e-input" placeholder="Type here.." />',
        okButton: { text: 'Connect'},
        cancelButton: {text:'Close'}
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/prompt-action-btn-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Show or hide dialog close button

When rendering the predefined dialogs through utility methods, You can close the dialog using the following ways. The default values of closeOnEscape and showCloseIcon is false.

- By pressing the escape key if the closeOnEscape property is enabled.
- By clicking the close button if the showCloseIcon property is enabled.

You can also manually close the Dialogs by creating an instance to the dialog and call the hide method.

Use the following code for **alert**, **confirm** and **prompt** to demonstrates the different ways of hiding the utility dialog.

Alert dialog close button

APP.VUE

```
<template>
  <div class="predefinedDialogs">
  <ejs-button id="alertDlgBtn" v-on:click.native="alertBtnClick"
cssClass="e-danger">Alert</ejs-button>
  </div>
</template>
<style>
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
    alertBtnClick: function () {
      DialogUtility.alert({
        title: "Low battery",
        content: "10% of battery remaining",
        width: "250px",
        showCloseIcon: true,
        closeOnEscape: true
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/alert-close-btn-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Confirm dialog close button

APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="confirmDlgBtn" v-on:click.native="confirmBtnClick"
cssClass="e-success">Confirm</ejs-button>
  </div>
</template>
<style>
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
    confirmBtnClick: function () {
      DialogUtility.confirm({
        title: 'Delete file',
        content: "Are you sure you want to permanently delete these items?",
        width: '300px',
        showCloseIcon: true,
        closeOnEscape: true
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/confirm-close-btn-cs1" %}

Prompt dialog close button

APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="promptDlgBtn" v-on:click.native="promptBtnClick"
:isPrimary="true">Prompt</ejs-button>
  </div>
</template>
<style>
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
```

```
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default {
  data: function () {
    return {};
  },
  methods: {
    promptBtnClick: function () {
      DialogUtility.confirm({
        title: "Join chat group",
        width: "300px",
        content:'<p>Enter your name: </p><input id= "inputEle" type="text"
name="Required" class="e-input" placeholder="Type here.." />',
        showCloseIcon: true,
        closeOnEscape: true
      });
    },
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/prompt-close-btn-cs1" %}

Customize dialog content

You can load custom content in predefined dialogs using the content property.

Use the following code to customize the dialog content to render the custom TextBox component inside the prompt dialog to get the username from the user.

APP.VUE

```
<template>
  <div class="predefinedDialogs">
    <ejs-button id="confirmDlgBtn" v-on:click.native="confirmBtnClick"
cssClass="e-success">Confirm </ejs-button>
  </div>
</template>
<style>
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
.predefinedDialogs {
  height: 100%;
  min-height: 350px;
}
</style>
<script>
import Vue from "vue";
import { DialogUtility } from "@syncfusion/ej2-vue-popups";
```

```
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(DialogUtility);
export default ( {
  data: function () {
    return {};
  },
  methods: {
    confirmBtnClick: function () {
      DialogUtility.confirm({
        title: "Join chat group",
        content: '<p>Enter your name: </p><input class="e-input"/>',
        width: "350px",
      });
    },
  },
});
</script>
```

{% previewsample "page.domainurl/code-snippet/predefined-dialogs/custom-content-cs1" %}

ProgressBar

Getting Started with the Vue Progressbar Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Progressbar component

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

Below is the list of minimum dependencies required to use the progressbar component.

`javascript

|-- @syncfusion/ej2-vue-progressbar

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-data: "*"

|-- @syncfusion/ej2-svg-base

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

Copyright © 2001 -2024 Syncfusion Inc.

`

or

`bash

yarn global add @vue/cli

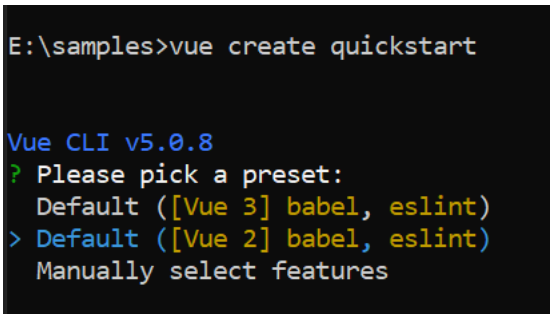
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Progressbar component as an example. Install the @syncfusion/ej2-vue-progressbar package by running the following command:

`bash

npm install @syncfusion/ej2-vue-progressbar --save

`

or

`bash

yarn add @syncfusion/ej2-vue-progressbar

`

The **--save** will instruct NPM to include the chart package inside of the dependencies section of the package.json.

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

Copyright © 2001 -2024 Syncfusion Inc.

The progressbar has different themes. They are:

- Material
- Fabric
- Bootstrap
- High Contrast

import chart component CSS as given below in <style> section of the App.vue file.

~/SRC/APP.VUE

```
<style>
<!-- Material theme used for this sample -->
@import "../node_modules/@syncfusion/ej2-vue-
progressbar/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Progressbar component:

1\. First, import and register the Progressbar component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { ProgressBarComponent } from '@syncfuion/ej2-vue-charts';
export default {
components: {
'ejs-progressbar': ProgressBarComponent
}
}
</script>
```

2\. In the template section, define the Progressbar component with the animation and value property.

~/SRC/APP.VUE

```
<template>
<div>
<div id='loader'>LOADING....</div>
<div id="container">
<div class="row linear-parent">
<div id="percentage" class="linear-progress">
<ejs-progressbar
id="percentage"
type='Circular'
:value='value'
:animation="animation"
>
</ejs-progressbar>
</div>
</div>
</div>
</div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

3\. Declare the value for the animation and value properties in the script section.

~/SRC/APP.VUE

```
<script>
data: function() {
return {
value : 100,
animation: {
enable: true,
duration: 2000,
delay: 0
},
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

~/SRC/APP.VUE

```
<template>
<div>
<div id='loader'>LOADING....</div>
<div id="container">
<div class="row linear-parent">
<div id="percentage" class="linear-progress">
<ejs-progressbar
id="percentage"
type='Circular'
:value='value'
:animation="animation"
>
</ejs-progressbar>
</div>
</div>
</div>
</div>
</template>
<script>
import { ProgressBarComponent } from "@syncfusion/ej2-vue-progressbar";
export default {
components: {
'ejs-progressbar': ProgressBarComponent
},
data: function() {
return {
value : 100,
animation: {
enable: true,
duration: 2000,
delay: 0
},
};
},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
provide: {},
methods: {}
}
</script>
<style>
#loader {
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
#container {
display: -webkit-box;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

Module Injection

Progressbar component are segregated into individual feature-wise modules. In order to use a particular feature, you need to inject its feature service in the AppModule. feature service name and description as follows.

- ProgressAnnotation - Inject this provider to use Annotation.

This module should be injected to the provide section as follows,

`javascript

import { ProgressBarComponent, ProgressAnnotationService } from "@syncfusion/ej2-vue-charts";

export default {

components: {

'ejs-progressbar': ProgressbarComponent

},

provide: {

Copyright © 2001 -2024 Syncfusion Inc.

progressbar: [ProgressAnnotationService]

}

};

`

Getting Started with the Vue Progress Bar Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Progress Bar component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Copyright © 2001 -2024 Syncfusion Inc.

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Progress Bar component as an example. To use the Vue Progress Bar component in the project, the @syncfusion/ej2-vue-progressbar package needs to be installed using the following command:

`bash

Copyright © 2001 -2024 Syncfusion Inc.

npm install @syncfusion/ej2-vue-progressbar --save

`

or

`bash

yarn add @syncfusion/ej2-vue-progressbar

`

Add Syncfusion Vue component

Follow the below steps to add the Vue Progress Bar component using Composition API or Options API:

1.First, import and register the Progress Bar component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { ProgressBarComponent as EjsProgressbar } from "@syncfusion/ej2-vue-progressbar";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { ProgressBarComponent } from "@syncfusion/ej2-vue-progressbar";
//Component registration
export default {
name: "App",
components: {
"ejs-progressbar": ProgressBarComponent
}
}
</script>
```

2.In the template section, define the Progress Bar component with the [dataSource] property.

~/SRC/APP.VUE

```
<template>
<ejs-progressbar id="percentage" :type='type'
:showProgressValue='showProgressValue' :value='value'
:animation="animation">
</ejs-progressbar>
</template>
```

3.Declare the values for the dataSource property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const type = 'Circular';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
const showProgressValue = 'true';
const value = 100;
const animation = {
enable: true,
duration: 2000,
delay: 0
};
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
type: 'Circular',
showProgressValue: 'true',
value: 100,
animation: {
enable: true,
duration: 2000,
delay: 0
}
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-progressbar id="percentage" :type='type'
:showProgressValue='showProgressValue' :value='value'
:animation="animation">
</ejs-progressbar>
</template>
<script setup>
import { ProgressBarComponent as EjsProgressbar } from "@syncfusion/ej2-vue-progressbar";
const type = 'Circular';
const showProgressValue = 'true';
const value = 100;
const animation = {
enable: true,
duration: 2000,
delay: 0
};
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-progressbar id="percentage" :type='type'
:showProgressValue='showProgressValue' :value='value'
:animation="animation">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</ejs-progressbar>
</template>
<script>
import { ProgressBarComponent } from "@syncfusion/ej2-vue-progressbar";
//Component registration
export default {
name: "App",
components: {
"ejs-progressbar": ProgressBarComponent
},
data() {
return {
type: 'Circular',
showProgressValue: 'true',
value: 100,
animation: {
enable: true,
duration: 2000,
delay: 0
}
};
},
};
</script>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

Sample: vue-3-progressbar-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

Copyright © 2001 -2024 Syncfusion Inc.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Types in Vue Progressbar component

Visualize progress in different shapes (rectangle, circle, and semi-circle) to give a unique appearance to your app design.

Linear

Set **type** to Linear to get the linear progress bar. It also support secondary progress and different mode of progress.

App.vue

`

```
<template>
<div id='loader'>LOADING....</div>
<div id='container'>
<div class="row linear-parent">
<div id="determinate" class="linear-progress">
<ejs-progressbar
id='determinate'
type='Linear'
height='60'
:value='value1'
:animation='animation'

</ejs-progressbar>
</div>
<div id="indeterminate" class="linear-progress">
<ejs-progressbar
id='indeterminate'
type='Linear'
height='60'
:value='value2'
:isIndeterminate='isIndeterminate'

</ejs-progressbar>
```

```
</div>
<div id="buffer" class="linear-progress">
<ejs-progressbar
id='buffer'
type='Linear'
height='60'
:value='value3'
:secondaryProgress='secondaryProgress'
:animation=' animation'

</ejs-progressbar>
</div>
<div id="segment" class="linear-progress">
<ejs-progressbar
id='segment'
type='Linear'
height='60'
:value='value4'
:segmentCount='count'
:animation='animation'

</ejs-progressbar>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
return {
value1:100,
value2:20,
value3:40,
value4:100,
isIndeterminate:true,
secondaryProgress:60,
animation: {
enable: true,
duration: 2000,
delay: 0
},
};
},
provide: {},
methods: {}
});
</script>
<style>
loader {
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
container {
display: -webkit-box;
}
</style>
`
```

Copyright © 2001 -2024 Syncfusion Inc.

Circular

Set **type** to Circular to get the circular progress bar. It also support secondary progress and different mode of progress.

App.vue

`

```
<template>
<div id='loader'>LOADING....</div>
<div id='container'>
<div class="row linear-parent">
<div id="determinate" class="linear-progress">
<ejs-progressbar
id='determinate'
type='Circular'
height='60'
:value='value1'
:animation='animation'

</ejs-progressbar>
</div>
<div id="indeterminate" class="linear-progress">
<ejs-progressbar
id='indeterminate'
type='Circular'
height='60'
:value='value2'
:isIndeterminate='isIndeterminate'

</ejs-progressbar>
</div>
<div id="buffer" class="linear-progress">
<ejs-progressbar
id='buffer'
type='Circular'
height='60'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
:value='value3'
:secondaryProgress='secondaryProgress'
:animation=' animation'

</ejs-progressbar>
</div>
<div id="segment" class="linear-progress">
<ejs-progressbar
id='segment'
type='Circular'
height='60'
:value='value4'
:segmentCount='count'
:animation='animation'

</ejs-progressbar>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
value1:100,
value2:20,
value3:40,
value4:100,
isIndeterminate:true,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
secondaryProgress:60,
animation: {
enable: true,
duration: 2000,
delay: 0
},
};
},
provide: {},
methods: {}
});
</script>
<style>
loader {
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
container {
display: -webkit-box;
}
</style>
`
```

Tooltip in Vue Progressbar component

Tooltip

The tooltip for the progress bar is used to represent the progress value. During the initial load, it can be enabled by using the enable property. The showTooltipOnHover property can show the tooltip on mouseover.

App.vue

```
`
<template>
```

```
<div id='loader'>LOADING....</div>
<div id='container'>
<div class="row linear-parent">
<div id="determinate" class="linear-progress">
<ejs-progressbar
id='determinate'
type='Linear'
height='60'
:value='value1'
:animation='animation'
:tooltip='tooltip'
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
value1:100,
animation: {
enable: true,
duration: 2000,
delay: 0
},
tooltip: {
enable: true,
showTooltipOnHover: true
}
```

Copyright © 2001 -2024 Syncfusion Inc.

};

},

provide: {},

methods: {}

});

</script>

<style>

loader {

color: #008cff;

height: 40px;

left: 45%;

position: absolute;

top: 45%;

width: 30%;

}

container {

display: -webkit-box;

}

</style>

`

Format

By default, the tooltip shows information about progress. In addition to that, show more information in the tooltip using the format property.

App.vue

`

<template>

LOADING....

<ejs-progressbar

id='determinate'

type='Linear'

height='60'

Copyright © 2001 -2024 Syncfusion Inc.

```
:value='value1'
:animation='animation'
:tooltip='tooltip'
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
value1:100,
animation: {
enable: true,
duration: 2000,
delay: 0
},
tooltip: {
enable: true,
format: "Progress: ${value}"
}
};
},
provide: {},
methods: {}
});
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

loader {

color: #008cff;

height: 40px;

left: 45%;

position: absolute;

top: 45%;

width: 30%;

}

container {

display: -webkit-box;

}

</style>

`

Customization

The fill and border properties are used to customize the background color and border of the tooltip respectively. The textStyle property in the tooltip is used to customize the font of the tooltip text.

App.vue

`

<template>

LOADING....

<ejs-progressbar

id='determinate'

type='Linear'

height='60'

:value='value1'

:animation='animation'

:tooltip='tooltip'

</template>

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
value1:100,
animation: {
enable: true,
duration: 2000,
delay: 0
},
tooltip: {
enable: true,
format: "Progress: ${value}"
}
};
},
provide: {},
methods: {}
});
</script>
<style>
```

loader {

```
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

container {

```
display: -webkit-box;
}
</style>
`
```

States in Vue Progressbar component

Visualize progress in different modes.

Determinate

This is the default state. You can use it when the progress estimation is known.

App.vue

```
`
<template>
<div id='loader'>LOADING....</div>
<div id='container'>
<div class="row linear-parent">
<div id="percentage" class="linear-progress">
<ejs-progressbar
id='percentage'
type='Linear'
height='60'
:value='value'

</ejs-progressbar>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
return {
value:100
};
},
provide: {},
methods: {}
});
</script>
<style>
loader {
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
container {
display: -webkit-box;
}
</style>
`
```

Indeterminate

By enabling the **IsIndeterminate** property, the state of the progress bar can be changed to indeterminate when the progress cannot be estimated or is not being calculated. It can be combined with determinate mode to know that the application is estimating progress before the actual progress starts.

App.vue

```
`
<template>
<div id='loader'>LOADING....</div>
<div id='container'>
<div class="row linear-parent">
<div id="percentage" class="linear-progress">
<ejs-progressbar
```

Copyright © 2001 -2024 Syncfusion Inc.

```
id='percentage'
type='Linear'
height='60'
:value='value'
:isIndeterminate='isIndeterminate'

</ejs-progressbar>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
value:20,
isIndeterminate:true
};
},
provide: {},
methods: {}
});
</script>
<style>
```

loader {

```
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
```

Copyright © 2001 -2024 Syncfusion Inc.

top: 45%;

width: 30%;

}

container {
display: -webkit-box;

}

</style>

`

Buffer

You can use a secondary progress indicator when the primary progress depends on the secondary progress. This will allow users to visualize both primary and secondary progress simultaneously.

App.vue

`

<template>

LOADING....

<ejs-progressbar

id='percentage'

type='Linear'

height='60'

:value='value'

:secondaryProgress='secondaryProgress'

</ejs-progressbar>

</template>

<script>

import Vue from "vue";

import { Browser } from "@syncfusion/ej2-base";

import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
value:40,
secondaryProgress:60
};
},
provide: {},
methods: {}
});
</script>
<style>
loader {
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
container {
display: -webkit-box;
}
</style>
`
```

Customization in Vue Progressbar component

Segments

We can divide a progress bar into multiple segments using a segmentCount to visualize the progress of multiple sequential tasks.

App.vue

```
`
<template>
<div id='loader'>LOADING....</div>
```

```
<div id='container'>
<div class="row linear-parent">
<div id="percentage" class="linear-progress">
<ejs-progressbar
id='percentage'
type='Circular'
height='60'
:segmentCount='segmentCount'
:value='value'
:animation='animation'

</ejs-progressbar>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
segmentCount:8,
value:100,
animation: {
enable: true,
duration: 2000,
delay: 0
},
};
},
```

Copyright © 2001 -2024 Syncfusion Inc.

provide: {},

methods: {}

});

</script>

<style>

loader {
color: #008cff;

height: 40px;

left: 45%;

position: absolute;

top: 45%;

width: 30%;

}

container {
display: -webkit-box;

}

</style>

s

`

Thickness

Customize the thickness of the track using trackThickness, progress using progressThickness and secondary progress using secondaryProgressThickness to render the progress bar with different appearances.

App.vue

`

<template>

LOADING....

<ejs-progressbar

id='percentage'

type='Linear'

height='60'

Copyright © 2001 -2024 Syncfusion Inc.

```
width='90%'
:trackThickness='trackThickness'
:secondaryProgressThickness='secondaryProgressThickness'
:progressThickness='progressThickness'
:value='value'
:animation='animation'

</ejs-progressbar>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
animation: {
enable: true,
duration: 2000,
delay: 0
},
trackThickness:24,
secondaryProgressThickness: 20,
progressThickness:24,
value:100
};
},
provide: {},
methods: {}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
});
</script>
<style>
loader {
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
container {
display: -webkit-box;
}
</style>
`
```

Radius

The radius of the progress bar can be customized using `radius` property and corner can be customized by **cornerRadius** property.

`App.vue`

```
`
<template>
<div id='loader'>LOADING....</div>
<div id='container'>
<div class="row linear-parent">
<div id="percentage" class="linear-progress">
<ejs-progressbar
id='percentage'
type='Circular'
height='160px'
width='160px'
trackColor='#FFD939'
radius='100%'
progressColor='white'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
cornerRadius='Round'
:trackThickness='trackThickness'
:progressThickness='progressThickness'
:value='value'
:animation='animation'>
</ejs-progressbar>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
animation: {
enable: true,
duration: 2000,
delay: 0
},
trackThickness:80,
progressThickness:10,
value:60
};
},
provide: {},
methods: {}
});
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
loader {
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
container {
display: -webkit-box;
}
</style>
`
```

InnerRadius

The inner radius of the progress bar can be customized using innerRadius property.

App.vue

```
`
<template>
<div id='loader'>LOADING....</div>
<div id='container'>
<div class="row linear-parent">
<div id="percentage" class="linear-progress">
<ejs-progressbar
id='percentage'
type='Circular'
height='160px'
width='160px'
trackColor='#FFD939'
radius='100%'
innerRadius='80%'
progressColor='white'
cornerRadius='Round'
:trackThickness='trackThickness'
:progressThickness='progressThickness'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
:value='value'
:animation='animation'>
</ejs-progressbar>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
animation: {
enable: true,
duration: 2000,
delay: 0
},
trackThickness:80,
progressThickness:10,
value:60
};
},
provide: {},
methods: {}
});
</script>
<style>
```

loader {

```
color: #008cff;
height: 40px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
container {
display: -webkit-box;
}
</style>
`
```

Progress color and track color

Customize the color of progress, secondary progress, and track by using the progressColor, secondaryProgressColor, and trackColor properties.

App.vue

```
`
<template>
<div id='loader'>LOADING....</div>
<div id='container'>
<div class="row linear-parent">
<div id="percentage" class="linear-progress">
<ejs-progressbar
id='percentage'
type='Linear'
height='60'
width='90%'
:trackThickness='trackThickness'
:progressThickness='progressThickness'
:secondaryProgress = '90'
:secondaryProgressColor='secondaryProgressColor'
:value='value'
progressColor='#E3165B'
trackColor='#F8C7D8'
:animation='animation'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</ejs-progressbar>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
animation: {
enable: true,
duration: 2000,
delay: 0
},
trackThickness:24,
progressThickness:24,
secondaryProgressColor: 'green',
value:70
};
},
provide: {},
methods: {}
});
</script>
<style>
```

loader {

```
color: #008cff;
height: 40px;
left: 45%;
```

Copyright © 2001 -2024 Syncfusion Inc.

position: absolute;

top: 45%;

width: 30%;

}

container {
display: -webkit-box;

}

</style>

`

Annotation in Vue Progressbar component

Annotation

In the circular progress bar, you can add any view to the center using the **Content** property in annotation.

For example, you can include add, start, or pause button to control the progress. You can also add an image that indicates the actual task in progress or add custom text that conveys how far the task is completed.

App.vue

`

<template>

LOADING....

<ejs-progressbar

id='percentage'

type='Circular'

:trackThickness='trackThickness'

:progressThickness='progressThickness'

<e-progressbar-annotations>

<e-progressbar-annotation

:content="content"

</e-progressbar-annotation>

</e-progressbar-annotations>

Copyright © 2001 -2024 Syncfusion Inc.

```
</ejs-progressbar>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin, ProgressAnnotation } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
trackThickness:10,
progressThickness:10,
content: '<div id="point1" style="font-size:20px;font-
weight:bold;color:#ffffff;fill:#ffffff"><span>60%span><div>'
};
},
provide: {
progressbar: [ProgressAnnotation]
},
methods: {}
});
</script>
<style>
```

loader {

```
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

container {

```
display: -webkit-box;
}
</style>
`
```

Label

You can show the progress value in both linear and cicular progress bar using **showProgressValue** property.

App.vue

```
`
<template>
<div id='loader'>LOADING....</div>
<div id='container'>
<div class="row linear-parent">
<div id="percentage" class="linear-progress">
<ejs-progressbar
id='percentage'
type='Linear'
:trackThickness='trackThickness'
:progressThickness='progressThickness'
:value='value'
:labelStyle='labelStyle'
:textRender='textRender'
:showProgressValue='showProgressValue'
:animation='animation'

</ejs-progressbar>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
trackThickness:24,
progressThickness:24,
value:50,
animation: {
enable: true,
duration: 2000,
delay: 0
},
labelStyle: {
color: '#FFFFFF'
},
showProgressValue: true
};
},
provide: {},
methods: {
textRender: function(args) {
args.text = '50';
}
}
});
</script>
<style>
```

loader {

```
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
```

Copyright © 2001 -2024 Syncfusion Inc.

top: 45%;

width: 30%;

}

container {
display: -webkit-box;

}

</style>

`

Animation in Vue Progressbar component

<!-- markdownlint-disable MD033 -->

Progress Bar support to animate the progress by using animation property. Enable the animation by setting **enable** property and also you can control the speed by using **duration** property.

App.vue

`

<template>

LOADING....

<ejs-progressbar

id="percentage"

type='Circular'

:value='value'

:animation="animation"

</ejs-progressbar>

</template>

<script>

import Vue from "vue";

import { Browser } from "@syncfusion/ej2-base";

import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
value : 100,
animation: {
enable: true,
duration: 2000,
delay: 0
},
};
},
provide: {},
methods: {}
});
</script>
<style>
loader {
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
container {
display: -webkit-box;
}
</style>
`
```

Range in Vue Progressbar component

<!-- markdownlint-disable MD033 -->

Range represents the entire span of the ProgressBar and can be defined using the `minimum` and `maximum` properties.

App.vue

```
<template>
<div id='loader'>LOADING....</div>
<div id='container'>
<div class="row linear-parent">
<div id="percentage" class="linear-progress">
<ejs-progressbar
id='percentage'
type='Linear'
:minimum='minimum'
:maximum='maximum'
:trackThickness='trackThickness'
:progressThickness='progressThickness'
:value='value'
:labelStyle='labelStyle'
:showProgressValue='showProgressValue'
:animation='animation'

</ejs-progressbar>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
trackThickness:24,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
progressThickness:24,
minimum:10,
maximum:90,
value:90,
animation: {
enable: true,
duration: 2000,
delay: 0
},
labelStyle: {
color: '#FFFFFF'
},
showProgressValue: true
};
},
provide: {},
methods: {}
});
</script>
<style>
loader {
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
container {
display: -webkit-box;
}
</style>
`
```

Copyright © 2001 -2024 Syncfusion Inc.

Events in Vue Progressbar component

Value Change

<!-- markdownlint-disable MD033 -->

valueChanged event is triggered when the progress value is changed.

App.vue

`

```
<template>
<div id='loader'>LOADING....</div>
<div id='container'>
<div class="row linear-parent">
<div id="percentage" class="linear-progress">
<ejs-progressbar
refs=event
id='percentage'
type='Linear'
:trackThickness='trackThickness'
:progressThickness='progressThickness'
:value='value'
:labelStyle='labelStyle'
:valueChanged='valueChanged'
:showProgressValue='showProgressValue'
:animation='animation'

</ejs-progressbar>
</div>
<button id="reLoad" @click="onClick()">ValueChanged</button>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
trackThickness:24,
progressThickness:24,
value:90,
animation: {
enable: true,
duration: 2000,
delay: 0
},
labelStyle: {
color: '#FFFFFF'
},
showProgressValue: true
};
},
provide: {},
methods: {
valueChanged: function(args) {
args.progressColor = '#2BB20E';
},
onClick: ()=> {
this.refs.event.value = 50;
}
}
});
</script>
<style>
```

loader {

```
color: #008cff;
height: 40px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
```

container {

```
display: -webkit-box;
}
</style>
`
```

ProgressCompleted

ProgressCompleted event is triggers when the ProgressBar attains the maximum value.

App.vue

```
`
<template>
<div id='loader'>LOADING....</div>
<div id='container'>
<div class="row linear-parent">
<div id="percentage" class="linear-progress">
<ejs-progressbar
id='percentage'
type='Linear'
:trackThickness='trackThickness'
:progressThickness='progressThickness'
:value='value'
:labelStyle='labelStyle'
:ProgressCompleted='ProgressCompleted'
:showProgressValue='showProgressValue'
:animation='animation'

</ejs-progressbar>
</div>
</div>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { Browser } from "@syncfusion/ej2-base";
import { ProgressBarPlugin } from "@syncfusion/ej2-vue-progressbar";
Vue.use(ProgressBarPlugin);
export default Vue.extend({
data: function() {
return {
trackThickness:24,
progressThickness:24,
value:90,
animation: {
enable: true,
duration: 2000,
delay: 0
},
labelStyle: {
color: '#FFFFFF'
},
showProgressValue: true
};
},
provide: {},
methods: {
ProgressCompleted: function(args) {
args.progressColor = '#2BB20E';
}
}
});
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

loader {

color: #008cff;

height: 40px;

left: 45%;

position: absolute;

top: 45%;

width: 30%;

}

container {

display: -webkit-box;

}

</style>

`

Accessibility in Vue Progress bar component

The Progress bar component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Progress bar component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/not-supported.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Progress bar component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Progress bar component:

- progressbar (role)
- aria-valuemin (attribute)
- aria-valuemax (attribute)
- aria-valuenow (attribute)
- aria-label (attribute)

Keyboard interaction

The Progress bar component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Progress bar component.

| **Press** | **To do this** |

| --- | --- |

| Tab | Moves the focus to the Progress bar element. |

| Ctrl + P | Prints the Progress bar. |

Ensuring accessibility

The Progress bar component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Progress bar component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Progress bar component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/progress-bar.html" %}

Copyright © 2001 -2024 Syncfusion Inc.

See also

- Accessibility in Syncfusion Vue components

ProgressButton

Getting Started with the Vue Progress button Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Progress button component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The list of dependencies required to use the ProgressButton component in your application is given as follows:

`js

|-- @syncfusion/ej2-vue-splitbuttons

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-splitbuttons

|-- @syncfusion/ej2-popups

|-- @syncfusion/ej2-buttons

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

vue create quickstart

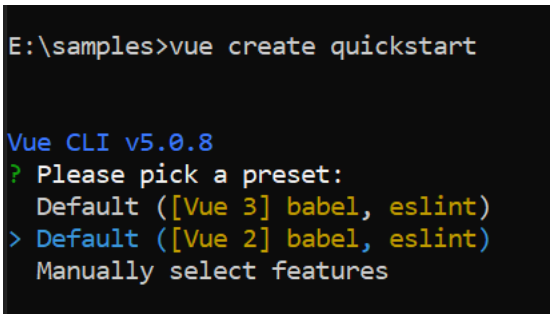
cd quickstart

yarn run serve

Copyright © 2001 -2024 Syncfusion Inc.

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Progress button component as an example. Install the @syncfusion/ej2-vue-splitbuttons package by running the following command:

`bash

npm install @syncfusion/ej2-vue-splitbuttons --save

`

or

`bash

yarn add @syncfusion/ej2-vue-splitbuttons

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Progress button component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Add Syncfusion Vue component

Follow the below steps to add the Vue Progress button component using Composition API or Options API:

1\. First, import and register the Progress button component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { ProgressButtonComponent as EjsProgressbutton } from
"@syncfusion/ej2-vue-splitbuttons";
</script>
```

COMPOSITION API (~/SRC/APP.VUE)

```
<script>
import { ProgressButtonComponent } from "@syncfusion/ej2-vue-splitbuttons";
export default {
components: {
'ejs-progressbutton': ProgressButtonComponent
}
}
</script>
```

2\. In the template section, define the Progress button component with the content property.

~/SRC/APP.VUE

```
<template>
<ejs-progressbutton content="Spin Left"></ejs-progressbutton>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
 <ejs-progressbutton content="Spin Left"></ejs-progressbutton>
</template>
<script setup>
import { ProgressButtonComponent as EjsProgressbutton } from
"@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css';
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<template>
 <ejs-progressbutton content="Spin Left"></ejs-progressbutton>
</template>
<script>
import { ProgressButtonComponent } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
  components: {
    'ejs-progressbutton': ProgressButtonComponent
  }
}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css';
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/progress-button/default-cs2" %}

ProgressButton supports different styles, types and sizes like Button. In addition, it also supports top and bottom icon positions.

Enable progress in button

You can enable the background filler UI by setting the enableProgress property to true.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
 <ejs-progressbutton content="Spin Left"  :enableProgress="true"></ejs-
progressbutton>
</template>
<script setup>
import { ProgressButtonComponent as EjsProgressbutton } from
"@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
enableRipple(true);
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css';
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
 <ejs-progressbutton content="Spin Left"  :enableProgress="true"></ejs-
progressbutton>
</template>
<script>
import { ProgressButtonComponent } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
  components: {
    'ejs-progressbutton': ProgressButtonComponent
  }
}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/progress-button/default-cs3" %}

See Also

- Spinner and Progress options

Getting Started with the Vue ProgressButton Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue ProgressButton component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Copyright © 2001 -2024 Syncfusion Inc.

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as `my-project` for this article.

`bash

? Project name: » my-project

`

2.Select `Vue` as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose `JavaScript` as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Copyright © 2001 -2024 Syncfusion Inc.

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue ProgressButton component as an example. To use the Vue ProgressButton component in the project, the @syncfusion/ej2-vue-splitbuttons package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-splitbuttons --save

`

or

`bash

yarn add @syncfusion/ej2-vue-splitbuttons

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the ProgressButton component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue ProgressButton component using Composition API or Options API:

1.First, import and register the ProgressButton component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { ProgressButtonComponent as EjsProgressbutton } from
"@syncfusion/ej2-vue-splitbuttons";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { ProgressButtonComponent } from "@syncfusion/ej2-vue-splitbuttons";
//Component registration
export default {
name: "App",
components: {
"ejs-progressbutton": ProgressButtonComponent
}
}
</script>
```

2.In the template section, define the ProgressButton component with the content property.

~/SRC/APP.VUE

```
<template>
<ejs-progressbutton content="content"></ejs-progressbutton>
</template>
```

3.Declare the values for the content property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const content = "Spin Left";
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
content = "Spin Left";
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-progressbutton content="content"></ejs-progressbutton>
</template>
<script setup>
import { ProgressButtonComponent as EjsProgressbutton } from
"@syncfusion/ej2-vue-splitbuttons";
const content = "Spin Left";
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-progressbutton content="Spin Left"></ejs-progressbutton>
</template>
<script>
import { ProgressButtonComponent } from "@syncfusion/ej2-vue-splitbuttons";
export default {
name: "App",
components: {
"ejs-progressbutton": ProgressButtonComponent
},
// Bound properties declarations
data() {
return {
content = "Spin Left";
};
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

Sample: vue-3-progressbutton-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

<!-- markdownlint-disable MD002 MD022 -->

Spinner and progress in Vue Progress button component

Change spinner position

Spinner position can be changed by modifying the position property ofspinSettingsModel. By default, the spinner is positioned at the left of the ProgressButton. You can position it at the left, right, top, bottom, or center of the text content.

Change spinner size

Spinner size can be changed by modifying the width property of spinSettingsModel. In this demo, the width is set to 20 to change the spinner size.

Spinner template

You can use custom spinner by specifying the template property of spinSettingsModel with custom styles.

The following sample demonstrates the above functionalities of the spinner.

APP.VUE

```
<template>
 <ejs-progressbutton content="Submit"  :spinSettings="spinSettings" ></ejs-
progressbutton>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { ProgressButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(ProgressButtonPlugin);
export default {
    data () {
        return {
            spinSettings: { position: 'Right', width: 20, template: '<div
class="template"></div>' }
        };
    }
}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @keyframes custom-rolling {
    0% {
      -webkit-transform: rotate(0deg);
      transform: rotate(0deg);
    }
    100% {
      -webkit-transform: rotate(360deg);
      transform: rotate(360deg);
    }
  }
  .template {
    border: 2px solid green;
    border-style: dotted;
    border-radius: 50%;
    border-top-color: transparent;
    border-bottom-color: transparent;
    height: 16px;
    width: 16px;
  }
  .template {
    -webkit-animation: custom-rolling 1.3s linear infinite;
    animation: custom-rolling 1.3s linear infinite;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/progress-button/default-cs6" %}

Progress

Content animation

The content of the ProgressButton can be animated during progress using the effect property of animationSettingsModel. You can also set custom duration and timing function using the duration and easing properties. The possible `effect` values are `None`, `SlideLeft`, `SlideRight`, `SlideUp`, `SlideDown`, `ZoomIn`, and `ZoomOut`.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
 <ejs-progressbutton content="Slide Left" :enableProgress="true"
:animationSettings="animationSettings" :spinSettings="spinSettings"></ejs-
progressbutton>
</template>
<script>
import Vue from 'vue';
import { ProgressButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(ProgressButtonPlugin);
export default {
    data () {
        return {
            animationSettings: { effect:'SlideLeft', duration: 500, easing:
'linear' },
            spinSettings: { position: 'Center' }
        };
    }
}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/progress-button/default-cs7" %}

Change step of the ProgressButton

The progress can be visualized at the specified interval by changing the step property in the begin event of the ProgressButton. In this demo, the step property is set to 20 to show progress at every 20% increment.

APP.VUE

```
<template>
 <ejs-progressbutton content='Progress Step' :enableProgress="true"
cssClass="e-hide-spinner" :begin="begin"></ejs-progressbutton>
</template>
<script>
import Vue from 'vue';
import { ProgressButtonPlugin} from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(ProgressButtonPlugin);
export default {
    methods: {
    begin: function(args) {
        args.step = 20;
    }
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/progress-button/default-cs8" %}

The class e-hide-spinner hides the spinner in the ProgressButton, For more information, see hide spinner section.

Change progress dynamically

The progress can be changed dynamically by modifying the percent property in the ProgressButton events. In this demo, on 40% completion of progress, the percent property is set to 90 to show dynamic change of the progress.

APP.VUE

```
<template>
<ejs-progressbutton :content="content" ref="progressbutton"
:enableProgress="true" duration=15000 :begin="begin" :end="end"
:progress="progress" cssClass="cssClass"></ejs-progressbutton>
</template>
<script>
import Vue from 'vue';
import { ProgressButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(ProgressButtonPlugin);
 export default {
    methods: {
    begin: function(args) {
        this.$refs.progressbutton.content = 'Progress ' + args.percent +
'%';
    },
    progress: function(args) {
       this.$refs.progressbutton.content = 'Progress ' + args.percent + '%';
       if(args.percent === 40) {
            args.percent = 90
        }
    },
    end: function(args) {
        this.$refs.progressbutton.content = 'Progress ' + args.percent +
'%';
    }
  },
  data () {
        return {
           content: "Progress Step",
           cssClass:   "e-hide-spinner"
    };
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/progress-button/default-cs9" %}

The method dataBind applies the property changes immediately to the component.

Start and stop methods

You can pause and resume the progress using the stop and start methods, respectively. In this demo, clicking the ProgressButton will pause and resume the progress.

APP.VUE

```
<template>
<ejs-progressbutton  ref="progressBtn" :content="content" :iconCss="iconCss"
:enableProgress="true" :cssClass="cssClass" v-on:click.native="clickHanlder"
:end="end"></ejs-progressbutton>
</template>
<script>
import Vue from 'vue';
import { ProgressButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(ProgressButtonPlugin);
var vue = export default {
 methods: {
    clickHanlder: function(event) {
        if(vue.content === 'Download') {
            vue.content = 'Pause';
            vue.iconCss = 'e-btn-sb-icon e-pause';
        }
        // clicking on ProgressButton will stop the progress when the text
content is 'Pause'
        else if(vue.content === 'Pause') {
            vue.content = 'Resume';
            vue.iconCss = 'e-btn-sb-icon e-play';
            this.$refs.progressBtn.stop();
        }
        // clicking on ProgressButton will start the progress when the text
content is 'Resume'
        else if(vue.content === 'Resume') {
            vue.content = 'Pause';
            vue.iconCss = 'e-btn-sb-icon e-pause';
            this.$refs.progressBtn.start();
        }
    },
    end: function(args) {
      vue.content = 'Download';
      vue.iconCss = 'e-btn-sb-icon e-download';
    }
  },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    data () {
        return {
            content: "Download",
            cssClass: "e-hide-spinner",
            iconCss: "e-btn-sb-icon e-download"
     };
   }
}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css';
  @font-face {
    font-family: 'btn-icon';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tSfgAAAEoAAAAVmNtYXDnH+dzAAABoAAAAEJnbHlm1v4
8pAAAAfgAAAQYaGVhZBOPfZcAAADQAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAAAYAAAAA
gbG9jYQN6ApQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYW1l07lFxAAABhAAAAIxcG9zdK9uovo
AAAhEAAAAgAABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAACAABAAAAAQAAJ1LUzF8
PPPUACwQAAAAAANg+nFMAAAAA2D6cUwAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAIAJ4AAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHA/QAAwAHAAAlIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgwBAgMFBQcICQkLCwwMDQ4NAtoNDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA0ODf0mDQ4NDAwLCwkJCAcFBQMCA239Jg4NDQ0LCwsJCQgHBQUDAgEBAgMFBQcICQkLCwsNDQ0
OAtoODQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEvBT8LLwg3NT8INS8FNT8NNS8JByU/BDUvCyMPAQytrQH5AgoEAQEBARg
hERESEyIJCSgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAgE
BAxYLBQQEAwMCAgIEBAoBAQEECgcHBgUFBAMDAQEBAQQFBwkFBQUGEf6tDwkEAwIBAQMDCgwVAwc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAgHCAcGBgYFBQkEAgYCBAwJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBwgHBgY
GBgoJCAYCAgEBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAgADAAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXMz8KNScxBxEfDjsBHQEfDTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFeBAMDAwIBAQGL7Y0EAwQCAgIBAYYKChEQDQsJCAcEBAU
CYt8BAQIDBAUFBQcHBwgICQgKjQECAgMEBAUFBgYHBgcIBwGcCAcHBwYGBgUFBAQDAgIBAQEBAgI
DBAQFBQYGBgcHBwgmAQMDAwUFBgYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAgMDAwTV+94BAQECAwMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBwYGBQQEAwI
CUAgHBwcGBgYFBQQEAwMBAgIBAwMEBAUFBQcGBwcHCAImCAcHBwYGBgUFBAQDAgIBAdUJCQgICAg
GBwYFBAQDAgEBAAAAAAIAAAAAA6cD9AADAAwAADchNSElAQcJAScBESNZA078sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAAAAAAAEAAAABAAAAAAABAAgAAQABAAAAAAA
CAAcACQABAAAAAAADAAgAEAABAAAAAAAAEAAgAGAABAAAAAAAFAAsAIAABAAAAAAAGAAgAKwABAAA
AAAAKACwAMwABAAAAAAALABIAXwADAAEECQAAAAIAcQADAAEECQABABAAcwADAAEECQACAA4AgwA
DAAEECQADABAAkQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
A1wADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bi1pY29uVmVyc2lvbiAxLjB
idG4taWNvbkZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMAbwBuAFIAZQBnAHUAbABhAHIAYgB0AG4ALQBpAGM
AbwBuAGIAdABuAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBpAGMAbwB
uAEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHM
AaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQB
vAG4ALgBjAG8AbQAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgBAgEDAQQBBQE
```

```
GAQcBCAEJAAptZWRpYS1wbGF5C21lZGlhLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AJbGlrZS0
tLTAxBGNvcHkQLWRvd25sb2FkLTAyLXdmLQAA) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  .e-btn-sb-icon {
    font-family: 'btn-icon' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
.e-download::before {
  content: '\e706';
}
.e-play::before {
  content: '\e700';
}
.e-pause::before {
  content: '\e701';
}
</style>
```

{% previewsample "page.domainurl/code-snippet/progress-button/default-cs10" %}

See Also

- How to hide spinner
- Customize ProgressButton using cssClass

Accessibility in Vue Progress button component

The Progress button component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Progress button component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Progress button component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Progress button component:

| Attributes | Purpose |

| --- | --- |

| aria-label | Provides an accessible name for the icon only Progress button. |

| aria-disabled | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Progress button component followed the [keyboard interaction] guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Progress button component.

| **Press** | **To do this** |

| --- | --- |

| Enter / Space | Starts the progress. |

Copyright © 2001 -2024 Syncfusion Inc.

Ensuring accessibility

The Progress button component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Progress button component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Progress button component with accessibility tools.

{% previewsample "page.domainurl/code-snippet/progress-button/default-cs1" %}

See also

- Accessibility in Syncfusion Vue components

Style and appearance in Vue Progress button component

To modify the ProgressButton appearance, you need to override the default CSS of ProgressButton component. Please find the list of CSS classes and its corresponding section in ProgressButton. Also, you have an option to create your own custom theme for the controls using our Theme Studio.

CSS Class | Purpose of Class

|.e-progress-btn|To customize the progress button

|.e-progress-btn:hover|To customize the progress button on hover

|.e-progress-btn:focus|To customize the progress button on focus

|.e-progress-btn .e-spinner-pane .e-spinner-inner svg .e-path-circle|To customize the progress button spinner

How To

Change the text content and styles of the progressbutton during progress in Vue Progress button component

You can change the text content and styles of the ProgressButton during progress by changing the text content and the cssClass property at the begin and end events.

APP.VUE

```
<template>
 <ejs-progressbutton ref="progressbutton" :content="content"  duration=4000
:enableProgress="true" :cssClass="cssClass" :begin ="begin"
:end="end"></ejs-progressbutton>
</template>
<script>
import Vue from 'vue';
import { ProgressButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple} from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(ProgressButtonPlugin);
export default {
methods: {
    begin: function(args) {
        this.$refs.progressbutton.content = 'Uploading...';
        this.$refs.progressbutton.cssClass = 'e-hide-spinner e-info';
    },
    end: function(args) {
```

```
        this.$refs.progressbutton.content = 'Success...';
        this.$refs.progressbutton.cssClass = 'e-hide-spinner e-success';
       setTimeout(()=>{
            this.$refs.progressbutton.content = 'Upload';
            this.$refs.progressbutton.cssClass = 'e-hide-spinner';
        }, 500)
   }
},
    data () {
        return {
           content: "Upload",
           cssClass:   "e-hide-spinner"
    };
  }
}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/progress-button/custom-progress-cs1" %}

Customize progress using cssclass in Vue Progress button component

You can customize the background filler UI using the cssClass property.

- Adding e-vertical to cssClass shows vertical progress.
- Adding e-progress-top to cssClass shows progress at the top.

You can also show reverse progress by adding custom class to the cssClass property.

APP.VUE

```
<template>
 <div>
 <ejs-progressbutton content='Vertical Progress' :enableProgress="true"
cssClass='e-hide-spinner e-vertical' duration=4000></ejs-progressbutton>
 <ejs-progressbutton content='Progress Top' :enableProgress="true"
cssClass='e-hide-spinner e-progress-top' duration=4000></ejs-progressbutton>
 <ejs-progressbutton content='Reverse Progress' :enableProgress="true"
cssClass='e-hide-spinner e-reverse-progress' duration=4000></ejs-
progressbutton>
 </div>
</template>
<script>
import Vue from 'vue';
import { ProgressButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(ProgressButtonPlugin);
export default {}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css';
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
button {
    margin: 25px;
}
.e-reverse-progress .e-progress {
    right: 0;
    left: auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/progress-button/custom-progress-cs2" %}

Hide spinner in Vue Progress button component

You can hide spinner in the ProgressButton by setting the e-hide-spinner property to cssClass.

APP.VUE

```
<template>
<ejs-progressbutton content="Progress" :enableProgress="true" cssClass='e-
hide-spinner'></ejs-progressbutton>
</template>
<script>
import Vue from 'vue';
import { ProgressButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(ProgressButtonPlugin);
export default {}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/progress-button/default-cs4" %}

Copyright © 2001 -2024 Syncfusion Inc.

Trace events of progress button in Vue Progress button component

The ProgressButton component triggers events based on its actions. The events can be used as extension points to perform custom operations.

The events available in ProgressButton are fail, begin, progress, and end.

APP.VUE

```
<template>
 <div id='container'>
        <div class="control-section">
            <div class="progress-btn-section">
                <ejs-progressbutton content='Progress'
:enableProgress="true" :begin="begin" :end="end" :progress="progress"
:fail="fail"></ejs-progressbutton>
            </div>
            <div class="property-section">
                <table id="propertyTable" title="Event trace">
                    <tbody>
                        <th>Event trace:-</th>
                        <tr>
                            <td></td>
                        </tr>
                    </tbody>
                </table>
            </div>
            <button id='clear' class='e-btn' @click='onClick'>Clear</button>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { ProgressButtonPlugin} from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(ProgressButtonPlugin);
export default {
    methods: {
        updateEventLog: function(args) {
            var propertyElem = document.getElementById('propertyTable');

propertyElem.getElementsByTagName('td')[0].insertAdjacentHTML('beforeend',
args.name + ' Event triggered. <br/>';
        },
        begin: function(args) {
            this.updateEventLog(args);
        },
        end: function(args) {
            this.updateEventLog(args);
        },
        progress: function(args) {
            this.updateEventLog(args);
        },
        fail: function(args) {
            this.updateEventLog(args);
        },
        onClick: function(args) {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            var propertyElem = document.getElementById('propertyTable');
            propertyElem.getElementsByTagName('td')[0].innerHTML = '';
        }
    }
}
</script>
<style>
    @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    .progress-btn-section {
        text-align: center;
        float: left;
    }
    .property-section {
        overflow: auto;
        width: 40%;
        height: 330px;
        float: right;
        font-family: monospace;
    }
    .property-section th {
        text-align: left;
    }
    #clear {
        float: right;
        clear: both;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/progress-button/default-cs5" %}

QueryBuilder

Getting Started with the Vue Query builder Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Query builder component.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The list of dependencies required to use the Query Builder component in your application is given below:

`js

|-- @syncfusion/ej2-vue-querybuilder

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-querybuilder

Copyright © 2001 -2024 Syncfusion Inc.

|-- @syncfusion/ej2-datamanager

|-- @syncfusion/ej2-dropdowns

|-- @syncfusion/ej2-calenders

|-- @syncfusion/ej2-inputs

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

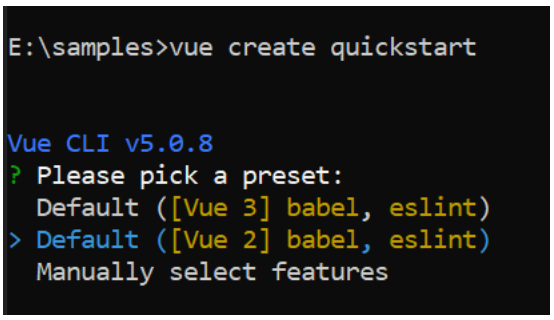
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Query builder component as an example. Install the @syncfusion/ej2-vue-querybuilder package by running the following command:

`bash

npm install @syncfusion/ej2-vue-querybuilder --save

`

or

`bash

yarn add @syncfusion/ej2-vue-querybuilder

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Query builder component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
</style>
```

Note: If you want to refer the combined component styles, please make use of our CRG (Custom Resource Generator) in your application.

Add Syncfusion Vue component

Follow the below steps to add the Vue Query builder component:

1\. First, import and register the Query builder component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { QueryBuilderComponent, ColumnDirective, ColumnsDirective } from
"@syncfusion/ej2-vue-querybuilder";
export default {
components: {
'ejs-querybuilder': QueryBuilderComponent,
'e-column': ColumnDirective,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
'e-columns': ColumnsDirective
}
}
</script>
```

2\. In the template section, define the Query builder component with the values property.

~/SRC/APP.VUE

```
<template>
<div class="control-section">
<div class="col-lg-12 querybuilder-control">
<ejs-querybuilder width="70%">
<e-columns>
<e-column field='EmployeeID' label='Employee ID' type='number' />
<e-column field='FirstName' label='First Name' type='string' />
<e-column field='TitleOfCourtesy' label='Title Of Courtesy' type='boolean'
:values="values" />
<e-column field='Title' label='Title' type='string' />
<e-column field='HireDate' label='Hire Date' type='date' format='dd/MM/yyyy'
/>
<e-column field='Country' label='Country' type='string' />
<e-column field='City' label='City' type='string' />
</e-columns>
</ejs-querybuilder>
</div>
</div>
</template>
```

3\. Declare the value for the values property in the script section.

~/SRC/APP.VUE

```
<script>
data: function() {
return {
values: ['Mr.', 'Mrs.']
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

~/SRC/APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%">
                <e-columns>
                    <e-column field='EmployeeID' label='Employee ID'
type='number' />
                    <e-column field='FirstName' label='First Name'
type='string' />
                    <e-column field='TitleOfCourtesy' label='Title Of
Courtesy' type='boolean' :values="values"/>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <e-column field='Title' label='Title' type='string' />
                    <e-column field='HireDate' label='Hire Date' type='date'
format='dd/MM/yyyy' />
                    <e-column field='Country' label='Country' type='string'
/>
                    <e-column field='City' label='City' type='string' />
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import { QueryBuilderComponent, ColumnDirective, ColumnsDirective } from
"@syncfusion/ej2-vue-querybuilder";
export default {
    components: {
        'ejs-querybuilder': QueryBuilderComponent,
        'e-column': ColumnDirective,
        'e-columns': ColumnsDirective
    },
    data: function() {
        return {
            values: ['Mr.', 'Mrs.']
        };
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs10" %}

Rendering with rule

Add the EJ2 Vue Query Builder using <ejs-querybuilder> to the <template> section of the App.vue file in src directory.

~/SRC/APP.VUE

```
<template>
<div class="control-section">
<div class="col-lg-12 querybuilder-control">
<ejs-querybuilder :dataSource="dataSource" :rule="importRules" width="70%">
<e-columns>
<e-column field='EmployeeID' label='Employee ID' type='number' />
<e-column field='FirstName' label='First Name' type='string' />
<e-column field='TitleOfCourtesy' label='Title Of Courtesy' type='boolean'
:values="values" />
<e-column field='Title' label='Title' type='string' />
<e-column field='HireDate' label='Hire Date' type='date' format='dd/MM/yyyy'
/>
<e-column field='Country' label='Country' type='string' />
<e-column field='City' label='City' type='string' />
</e-columns>
</ejs-querybuilder>
</div>
</div>
</template>
<style>
.e-query-builder {
margin: 0 auto;
}
</style>
<script>
import { QueryBuilderComponent, ColumnDirective, ColumnsDirective } from
"@syncfusion/ej2-vue-querybuilder";
export default {
components: {
'ejs-querybuilder': QueryBuilderComponent,
'e-column': ColumnDirective,
'e-columns': ColumnsDirective
},
data: function() {
return {
dataSource: employeeData,
values: ['Mr.', 'Mrs.'],
importRules: {
'condition': 'and',
'rules': [{
'label': 'Employee ID',
'field': 'EmployeeID',
'type': 'number',
'operator': 'equal',
'value': 1
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
{
'label': 'Title',
'field': 'Title',
'type': 'string',
'operator': 'equal',
'value': 'Sales Manager'
}]
}
};
}
}
var employeeData = [{
'EmployeeID': 1,
'FirstName': 'Nancy',
'Title': 'Sales Representative',
'TitleOfCourtesy': 'Ms.',
'HireDate': '22/07/2001',
'City': 'Seattle',
'Country': 'USA'
},
{
'EmployeeID': 2,
'FirstName': 'Andrew',
'Title': 'Vice President',
'TitleOfCourtesy': 'Dr.',
'HireDate': '21/04/2003',
'City': 'Tacoma',
'Country': 'USA'
},
{
'EmployeeID': 3,
'FirstName': 'Janet',
'Title': 'Sales Representative',
'TitleOfCourtesy': 'Ms.',
'HireDate': '22/07/2001',
'City': 'Kirkland',
'Country': 'USA'
}];
</script>
```

The following example shows a basic Query builder.

~/SRC/APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="dataSource"
:rule="importRules">
                <e-columns>
                    <e-column field='EmployeeID' label='Employee ID'
type='number' />
                    <e-column field='FirstName' label='First Name'
type='string' />
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <e-column field='TitleOfCourtesy' label='Title Of
Courtesy' type='boolean' :values="values"/>
                    <e-column field='Title' label='Title' type='string' />
                    <e-column field='HireDate' label='Hire Date' type='date'
format='dd/MM/yyyy' />
                    <e-column field='Country' label='Country' type='string'
/>
                    <e-column field='City' label='City' type='string' />
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import { QueryBuilderComponent, ColumnDirective, ColumnsDirective } from
"@syncfusion/ej2-vue-querybuilder";
export default {
    components: {
        'ejs-querybuilder': QueryBuilderComponent,
        'e-column': ColumnDirective,
        'e-columns': ColumnsDirective
    },
    data: function() {
        return {
            dataSource: employeeData,
            values: ['Mr.', 'Mrs.'],
            importRules: {
                'condition': 'and',
                'rules': [{
                    'label': 'Employee ID',
                    'field': 'EmployeeID',
                    'type': 'number',
                    'operator': 'equal',
                    'value': 1
                },
                {
                    'label': 'Title',
                    'field': 'Title',
                    'type': 'string',
                    'operator': 'equal',
                    'value': 'Sales Manager'
                }]
            }
        };
    }
}
var employeeData = [{
      'EmployeeID': 1,
      'FirstName': 'Nancy',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Seattle',
      'Country': 'USA'
    },
    {
      'EmployeeID': 2,
```

```
      'FirstName': 'Andrew',
      'Title': 'Vice President',
      'TitleOfCourtesy': 'Dr.',
      'HireDate': '21/04/2003',
      'City': 'Tacoma',
      'Country': 'USA'
    },
    {
      'EmployeeID': 3,
      'FirstName': 'Janet',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Kirkland',
      'Country': 'USA'
    }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs11" %}

You can also explore our Vue Query Builder example that shows how to render the Query Builder in Vue.

Columns in Vue Query builder component

The column definitions are used as the dataSource schema in the Query Builder. This plays a vital role in rendering column values. The query builder operations such as create or delete conditions and create or delete group they are performed based on the column definitions. The field property of the columns is necessary to map the data source values in the query builder columns.

If the column field is not specified in the dataSource, the column values will be empty.

Auto generation

The columns are automatically generated when the columns declaration is empty or undefined while initializing the query builder. All the columns in the dataSource are bound as the query builder columns.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="dataSource">
```

```
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { employeeData } from './datasource.js';
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: employeeData,
            values: ['Mr.', 'Mrs.']
        };
    }
}
var employeeData = [{
      'EmployeeID': 1,
      'FirstName': 'Nancy',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Seattle',
      'Country': 'USA'
    },
    {
      'EmployeeID': 2,
      'FirstName': 'Andrew',
      'Title': 'Vice President',
      'TitleOfCourtesy': 'Dr.',
      'HireDate': '21/04/2003',
      'City': 'Tacoma',
      'Country': 'USA'
    },
    {
      'EmployeeID': 3,
      'FirstName': 'Janet',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Kirkland',
      'Country': 'USA'
    }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs1" %}

When columns are auto-generated, the column type will be determined from the first record of the dataSource.

Labels

By default, the column label is displayed from the column field value. To override the default label, you have to define the label value.

Operators

The operator for a column can be defined in the operators property.

The available operators and its supported data types are:

| Operators | Description | Supported Types |

| ---------- | ---------- | ---------- |

| startswith | Checks whether the value begins with the specified value. | String |

| endswith | Checks whether the value ends with the specified value. | String |

| contains | Checks whether the value contains the specified value. | String |

| equal | Checks whether the value is equal to the specified value. | String Number Date Boolean |

| notequal | Checks whether the value is not equal to the specified value. | String Number Date Boolean |

| greaterthan | Checks whether the value is greater than the specified value. | Date Number |

| greaterthanorequal | Checks whether a value is greater than or equal to the specified value. | Date Number |

| lessthan | Checks whether the value is less than the specified value.| Date Number |

| lessthanorequal | Checks whether the value is less than or equal to the specified value. | Date Number |

| between | Checks whether the value is between the two-specific value. | Date Number |

| notbetween | Checks whether the value is not between the two-specific value. | Date Number |

| in | Checks whether the value is one of the specific values. | String Number |

| notin | Checks whether the value is not in the specific values. | String Number |

Step

The Query Builder allows you to set the step values to the number fields. So that you can easily access the numeric textbox. Use the step property, to set the step value for number values.

Format

The Query Builder formats date and number values. Use the format property to format date and number values.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="dataSource"
:rule="importRules">
                <e-columns>
                    <e-column field='EmployeeID' label='Employee ID'
type='number' step=10 :operators="employeeOperators" />
                    <e-column field='FirstName' label='First Name'
type='string' />
                    <e-column field='TitleOfCourtesy' label='Title Of
Courtesy' type='boolean' :values="values"/>
                    <e-column field='Title' label='Title' type='string' />
                    <e-column field='HireDate' label='Hire Date' type='date'
format='dd/MM/yyyy' />
                    <e-column field='Country' label='Country' type='string'
/>
                    <e-column field='City' label='City' type='string' />
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: employeeData,
            values: ['Mr.', 'Mrs.'],
            importRules: {
        'condition': 'and',
        'rules': [{
            'label': 'Employee ID',
            'field': 'EmployeeID',
            'type': 'number',
            'operator': 'equal',
            'value': 1001
        },
        {
            'label': 'Hire Date',
            'field': 'HireDate',
            'type': 'date',
            'operator': 'equal',
            'value': '07/05/1991'
        }]
    },
     employeeOperators:  [
        { value: 'equal', key: 'Equal' },
        { value: 'notequal', key: 'Not Equal' },
        { value: 'in', key: 'In' },
        { value: 'notin', key: 'Not In' }
    ]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        };
    }
}
var employeeData = [{
      'EmployeeID': 1,
      'FirstName': 'Nancy',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Seattle',
      'Country': 'USA'
    },
    {
      'EmployeeID': 2,
      'FirstName': 'Andrew',
      'Title': 'Vice President',
      'TitleOfCourtesy': 'Dr.',
      'HireDate': '21/04/2003',
      'City': 'Tacoma',
      'Country': 'USA'
    },
    {
      'EmployeeID': 3,
      'FirstName': 'Janet',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Kirkland',
      'Country': 'USA'
    }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs2" %}

Validations

Validation allows you to validate the conditions and it display errors for invalid fields while using the validateFields method. To enable validation in the query builder , set the allowValidation to true. Column fields are validated after setting allowValidation as to true. So, you should manually configure the validation for Operator and, Value fields through validation.

Copyright © 2001 -2024 Syncfusion Inc.

Install Syncfusion Buttons packages using below command.

`bash

npm install @syncfusion/ej2-vue-buttons --save

`

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder ref="querybuilder" width="70%"
:dataSource="dataSource" allowValidation="true">
                <e-columns>
                    <e-column field='EmployeeID' label='Employee ID'
type='number' validation="validateRule" />
                    <e-column field='FirstName' label='First Name'
type='string' />
                    <e-column field='TitleOfCourtesy' label='Title Of
Courtesy' type='boolean' :values="values"/>
                    <e-column field='Title' label='Title' type='string'
validation="validateRule" />
                    <e-column field='HireDate' label='Hire Date' type='date'
format='dd/MM/yyyy'  />
                    <e-column field='Country' label='Country' type='string'
validation="validateRule"/>
                    <e-column field='City' label='City' type='string'
validation="validateRule"/>
                </e-columns>
            </ejs-querybuilder>
             <ejs-button cssClass="e-qb-button" :isPrimary="true" v-
on:click.native="btnClick">Validate Fields</ejs-button>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: employeeData,
             values: ['Mr.', 'Mrs.'],
            validateRule: { isRequired: true }
        };
    },
    methods: {
    btnClick: function(event) {
     this.$refs.querybuilder.$el.ej2_instances[0].validateFields();
    }
  }
}
var employeeData = [{
```

```
      'EmployeeID': 1,
      'FirstName': 'Nancy',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Seattle',
      'Country': 'USA'
    },
    {
      'EmployeeID': 2,
      'FirstName': 'Andrew',
      'Title': 'Vice President',
      'TitleOfCourtesy': 'Dr.',
      'HireDate': '21/04/2003',
      'City': 'Tacoma',
      'Country': 'USA'
    },
    {
      'EmployeeID': 3,
      'FirstName': 'Janet',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Kirkland',
      'Country': 'USA'
    }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
    .e-qb-button {
        margin: 2% 1% 0 15%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs3" %}

Set isRequired validation for `Operator` and `Value` fields.

Set max, min values for number values.

Copyright © 2001 -2024 Syncfusion Inc.

Data binding in Vue Query builder component

The Query Builder uses DataManager, which supports both RESTful JSON data services binding and local JavaScript object array binding. The dataSource property can be assigned either with the instance of DataManager or JavaScript object array collection. It supports two kinds of data binding method:

- Local data
- Remote data

Local data

To bind local data to the query builder, you can assign the dataSource property with a JavaScript object array. The local data source can also be provided as an instance of the DataManager.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="dataSource"
:rule="importRules">
                <e-columns>
                    <e-column field='EmployeeID' label='Employee ID'
type='number' />
                    <e-column field='FirstName' label='First Name'
type='string' />
                    <e-column field='TitleOfCourtesy' label='Title Of
Courtesy' type='boolean' :values="values"/>
                    <e-column field='Title' label='Title' type='string' />
                    <e-column field='HireDate' label='Hire Date' type='date'
format='dd/MM/yyyy' />
                    <e-column field='Country' label='Country' type='string'
/>
                    <e-column field='City' label='City' type='string' />
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: employeeData,
            values: ['Mr.', 'Mrs.'],
            importRules: {
                'condition': 'and',
                'rules': [{
                    'label': 'Employee ID',
                    'field': 'EmployeeID',
                    'type': 'number',
                    'operator': 'equal',
                    'value': 1
                },
```

```
                {
                    'label': 'Title',
                    'field': 'Title',
                    'type': 'string',
                    'operator': 'equal',
                    'value': 'Sales Manager'
                }]
            }
        };
    }
}
var employeeData = [{
      'EmployeeID': 1,
      'FirstName': 'Nancy',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Seattle',
      'Country': 'USA'
    },
    {
      'EmployeeID': 2,
      'FirstName': 'Andrew',
      'Title': 'Vice President',
      'TitleOfCourtesy': 'Dr.',
      'HireDate': '21/04/2003',
      'City': 'Tacoma',
      'Country': 'USA'
    },
    {
      'EmployeeID': 3,
      'FirstName': 'Janet',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Kirkland',
      'Country': 'USA'
    }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs4" %}

By default, DataManager uses JsonAdaptor for local data-binding.

Remote data

To bind remote data to the query builder, assign service data as an instance of DataManager to the dataSource property. To interact with remote data source, provide the endpoint url.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="data"
:rule="importRules">
                <e-columns>
                    <e-column field='EmployeeID' label='Employee ID'
type='number' />
                    <e-column field='FirstName' label='First Name'
type='string' />
                    <e-column field='TitleOfCourtesy' label='Title Of
Courtesy' type='boolean' :values="values"/>
                    <e-column field='Title' label='Title' type='string' />
                    <e-column field='HireDate' label='Hire Date' type='date'
format='dd/MM/yyyy' />
                    <e-column field='Country' label='Country' type='string'
/>
                    <e-column field='City' label='City' type='string' />
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { DataManager, ODataAdaptor } from "@syncfusion/ej2-data";
Vue.use(QueryBuilderPlugin);
export default {
   data(){
        return {
            values: ['Mr.', 'Mrs.'],
            data: new DataManager({
            url:
"https://services.syncfusion.com/js/production/api/Employees/";
            adaptor: new ODataAdaptor(),
          }),
          importRules: {
                'condition': 'and',
                'rules': [{
                    'label': 'Employee ID',
                    'field': 'EmployeeID',
                    'type': 'number',
                    'operator': 'equal',
                    'value': 1
                },
                {
```

```
                    'label': 'Title',
                    'field': 'Title',
                    'type': 'string',
                    'operator': 'equal',
                    'value': 'Sales Manager'
                }]
            }
        };
    };
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs5" %}

By default, DataManager uses ODataAdaptor for remote data-binding.

Binding with OData services

OData is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the DataManager. Refer to the following code example for remote Data binding using OData service.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="data"
:rule="importRules">
                <e-columns>
                    <e-column field='EmployeeID' label='Employee ID'
type='number' />
                    <e-column field='FirstName' label='First Name'
type='string' />
                    <e-column field='TitleOfCourtesy' label='Title Of
Courtesy' type='boolean' :values="values"/>
                    <e-column field='Title' label='Title' type='string' />
                    <e-column field='HireDate' label='Hire Date' type='date'
format='dd/MM/yyyy' />
                    <e-column field='Country' label='Country' type='string'
/>
                    <e-column field='City' label='City' type='string' />
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
```

```
import { DataManager, ODataAdaptor } from "@syncfusion/ej2-data";
Vue.use(QueryBuilderPlugin);
export default {
    data(){
        return {
            values: ['Mr.', 'Mrs.'],
            data: new DataManager({
            url:
"https://services.syncfusion.com/js/production/api/Employees";
            adaptor: new ODataAdaptor(),
            crossDomain: true
          }),
            importRules: {
                'condition': 'and',
                'rules': [{
                    'label': 'Employee ID',
                    'field': 'EmployeeID',
                    'type': 'number',
                    'operator': 'equal',
                    'value': 1
                },
                {
                    'label': 'Title',
                    'field': 'Title',
                    'type': 'string',
                    'operator': 'equal',
                    'value': 'Sales Manager'
                }]
            }
        };
    };
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs6" %}

Binding with OData v4 services

The ODataV4 is an improved version of OData protocols, and the DataManager can also retrieve and consume OData v4 services. For more details on OData v4 services, refer to the odata documentation. To bind OData v4 service, use the ODataV4Adaptor.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="data"
:rule="importRules">
                <e-columns>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <e-column field='EmployeeID' label='Employee ID'
type='number' />
                    <e-column field='FirstName' label='First Name'
type='string' />
                    <e-column field='TitleOfCourtesy' label='Title Of
Courtesy' type='boolean' :values="values"/>
                    <e-column field='Title' label='Title' type='string' />
                    <e-column field='HireDate' label='Hire Date' type='date'
format='dd/MM/yyyy' />
                    <e-column field='Country' label='Country' type='string'
/>
                    <e-column field='City' label='City' type='string' />
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { DataManager, ODataV4Adaptor } from "@syncfusion/ej2-data";
Vue.use(QueryBuilderPlugin);
export default {
    data(){
        return {
            values: ['Mr.', 'Mrs.'],
            data: new DataManager({
            url:
"https://services.odata.org/V4/Northwind/Northwind.svc/Orders/";
            adaptor: new ODataV4Adaptor(),
            crossDomain: true
          }),
            importRules: {
                'condition': 'and',
                'rules': [{
                    'label': 'Employee ID',
                    'field': 'EmployeeID',
                    'type': 'number',
                    'operator': 'equal',
                    'value': 1
                },
                {
                    'label': 'Title',
                    'field': 'Title',
                    'type': 'string',
                    'operator': 'equal',
                    'value': 'Sales Manager'
                }]
            }
        };
    };
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs7" %}

Web API

You can use WebApiAdaptor to bind query builder with Web API created using OData endpoint.

`

```
<template>
<div class="control-section">
<div class="col-lg-12 querybuilder-control">
<ejs-querybuilder width="70%" :dataSource="data" :rule="importRules">
<e-columns>
<e-column field='EmployeeID' label='Employee ID' type='number' />
<e-column field='FirstName' label='First Name' type='string' />
<e-column field='TitleOfCourtesy' label='Title Of Courtesy' type='boolean' :values="values"/>
<e-column field='Title' label='Title' type='string' />
<e-column field='HireDate' label='Hire Date' type='date' format='dd/MM/yyyy' />
<e-column field='Country' label='Country' type='string' />
<e-column field='City' label='City' type='string' />
</e-columns>
</ejs-querybuilder>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { DataManager, WebApiAdaptor } from "@syncfusion/ej2-data";
Vue.use(QueryBuilderPlugin);
export default {
data: function() {
return {
values: ['Mr.', 'Mrs.'],
data: new DataManager({
```

Copyright © 2001 -2024 Syncfusion Inc.

```
url: 'api/OrderAPI',
adaptor: new WebApiAdaptor(),
crossDomain: true
}),
importRules: {
'condition': 'and',
'rules': [{
'label': 'Employee ID',
'field': 'EmployeeID',
'type': 'number',
'operator': 'equal',
'value': 1
},
{
'label': 'Title',
'field': 'Title',
'type': 'string',
'operator': 'equal',
'value': 'Sales Manager'
}]
}
};
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-querybuilder/styles/material.css";
.e-query-builder {
margin: 0 auto;
}
</style>
`
```

Copyright © 2001 -2024 Syncfusion Inc.

Data Manager

You can use the created conditions in DataManager through the getPredicate method. This method creates predicates which is used as conditions in DataManager.

Install Syncfusion Buttons packages using below command.

`bash

npm install @syncfusion/ej2-vue-buttons --save

`

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder ref="querybuilder" width="70%"
:dataSource="dataSource" :rule="importRules">
                <e-columns>
                    <e-column field='EmployeeID' label='Employee ID'
type='number' />
                    <e-column field='FirstName' label='First Name'
type='string' />
                    <e-column field='TitleOfCourtesy' label='Title Of
Courtesy' type='boolean' :values="values"/>
                    <e-column field='Title' label='Title' type='string' />
                    <e-column field='HireDate' label='Hire Date' type='date'
format='dd/MM/yyyy' />
                    <e-column field='Country' label='Country' type='string'
/>
                    <e-column field='City' label='City' type='string' />
                </e-columns>
            </ejs-querybuilder>
             <ejs-button cssClass="e-qb-button" :isPrimary="true" v-
on:click.native="btnClick">Get Data</ejs-button>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { DataManager, Query } from '@syncfusion/ej2-data';
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { Compile } from '@syncfusion/ej2-base'
Vue.use(ButtonPlugin);
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: employeeData,
            values: ['Mr.', 'Mrs.'],
            importRules: {
                'condition': 'and',
                'rules': [{
                    'label': 'Employee ID',
                    'field': 'EmployeeID',
                    'type': 'number',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    'operator': 'equal',
                    'value': 1
                }]
            }
        };
    },
      methods: {
    btnClick: function(event) {
        var validRule =
this.$refs.querybuilder.ej2Instances.getValidRules(this.$refs.querybuilder.e
j2Instances.rule);
        var predicate =
this.$refs.querybuilder.ej2Instances.getPredicate(validRule);
        var dataManagerQuery = new Query().select(['EmployeeID', 'Title',
'City']);
        var fltrDataSource = [];
          new DataManager(employeeData).executeQuery(dataManagerQuery)
            .then((e) => {
                (e.result).forEach((data) => {
                      fltrDataSource.push(data);
                    //recieve the result data
                });
            });
    }
  }
}
var employeeData = [{
      'EmployeeID': 1,
      'FirstName': 'Nancy',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Seattle',
      'Country': 'USA'
    },
    {
      'EmployeeID': 2,
      'FirstName': 'Andrew',
      'Title': 'Vice President',
      'TitleOfCourtesy': 'Dr.',
      'HireDate': '21/04/2003',
      'City': 'Tacoma',
      'Country': 'USA'
    },
    {
      'EmployeeID': 3,
      'FirstName': 'Janet',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Kirkland',
      'Country': 'USA'
    }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
    .e-qb-button {
        margin: 2% 1% 0 15%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs8" %}

Complex Data Binding

Complex Data Binding allows you to create subfield for columns. To implement complex data binding, either bind the complex data in nested columns or specify complex data source and separator must be given in querybuilder.

In the following sample, complex data was bound in nested columns.

APP.VUE

```
<template>
<div class="control-section">
<div>
    <table>
        <tr>
            <td> <ejs-button id="reset" class="e-control e-danger e-btn e-
small"  v-on:click.native='onReset'>Reset</ejs-button> </td>
            <td> <ejs-button id="rule" class="e-control e-success e-btn e-
small"  v-on:click.native='onSetsqlRules'>SetSqlRules</ejs-button> </td>
            <td> <ejs-button id="sql" class="e-control e-success e-btn e-
small"  v-on:click.native='onSetrules' >SetRules</ejs-button> </td>
        </tr>
    </table>
</div>
 <ejs-querybuilder id="querybuilder" ref="querybuilder" :rule="importRules"
separator =".". enableNotCondition = true>
                <e-columns>
                    <e-column field="Employee" label="Employee"
:columns='columns1'/>
                    <e-column field="Name" label="Name" :columns="columns2"
/>
                    <e-column field='Country' label='Country'
:columns="columns3"/>
                </e-columns>
</ejs-querybuilder>
</div>
</template>
<script>
```

```
import Vue from "vue";
import { QueryBuilderPlugin } from '@syncfusion/ej2-vue-querybuilder';
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { getComponent, compile,closest } from '@syncfusion/ej2-base';
import { CheckBoxPlugin, ButtonPlugin, RadioButtonPlugin } from
"@syncfusion/ej2-vue-buttons";
import { DropDownButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
Vue.use(QueryBuilderPlugin);
Vue.use(DropDownListPlugin);
Vue.use(CheckBoxPlugin);
Vue.use(DropDownButtonPlugin);
Vue.use(ButtonPlugin);
Vue.use(RadioButtonPlugin);
export default {
    data: function() {
        return {
            importRules: {
                condition: 'and',
                rules: [{
                    label: 'ID',
                    field: 'Employee.ID',
                    type: 'string',
                    operator: 'equal',
                    value: 0
                },
                {
                    label: 'Last Name',
                    field: 'Name.LastName',
                    type: 'string',
                    operator: 'contains',
                    value: 'malan'
                },
                {
                    condition: 'or',
                    rules: [{
                        label: 'City',
                        field: 'Country.State.City',
                        operator: 'startswith',
                        type: 'string',
                        value: 'U'
                    },
                    {
                        label: 'Region',
                        field: 'Country.Region',
                        operator: 'endswith',
                        type: 'string',
                        value: 'c'
                    },
                    {
                        label: 'Name',
                        field: 'Country.Name',
                        operator: 'isnotempty'
                    }]
                }]
            },
            columns1: [
                { field: 'ID', label: 'ID', type: 'number'},
```

```
                { field: 'DOB', label: 'Date of birth', type: 'date'},
                { field: 'HireDate', label: 'Hire Date', type: 'date'},
                { field: 'Salary', label: 'Salary', type: 'number'},
                { field: 'Age', label: 'Age', type: 'number'},
                { field: 'Title', label: 'Title', type: 'string'}
            ],
            columns2: [
                { field: 'FirstName', label: 'First Name', type: 'string'},
                { field: 'LastName', label: 'Last Name', type: 'string'}
            ],
            columns3: [
                { field: 'State', label: 'State', columns : [
                    { field: 'City', label: 'City', type: 'string'},
                    { field: 'Zipcode', label: 'Zip Code', type: 'number'}]
},
                { field: 'Region', label: 'Region', type: 'string'},
                { field: 'Name', label: 'Name', type: 'string'}
            ],
        };
    },
     methods: {
        onReset: function(event){
            this.$refs.querybuilder.ej2Instances.reset();
        },
        onSetsqlRules: function(event){

this.$refs.querybuilder.ej2Instances.setRulesFromSql("Employee.ID = 0 AND
Name.LastName LIKE ('%malan%') AND (Country.State.City LIKE ('U%') AND
Country.Region LIKE ('%c') AND Country.Name IS NOT EMPTY)");
        },
        onSetrules: function(event){
            this.$refs.querybuilder.ej2Instances.setRules(this.importRules);
        }
    }
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder .e-group-body .e-horizontal-mode .e-rule-sub-filter{
        display: inline-block;
    }
    .e-query-builder .e-group-body .e-rule-container .e-rule-sub-filter{
        padding: 12px 0 12px 12px;
        width: auto;
    }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/query-builder/complex-data-binding-cs1" %}

Filtering in Vue Query builder component

Query Builder allows you to create or delete conditions and groups. You can use showButtons to enable/disable these buttons.

You can create or delete conditions by interacting through the user interface and methods.

- Use the addRules, and deleteRules methods to create/delete conditions.
- Use addGroups, and deleteGroups methods to create/delete groups.

Install Syncfusion Buttons packages using below command.

`bash

npm install @syncfusion/ej2-vue-buttons --save

`

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" id="querybuilder"
ref="querybuilder" :dataSource="dataSource" :rule="importRules"
:showButtons="showButtons">
                <e-columns>
                    <e-column field='EmployeeID' label='Employee ID'
type='number' />
                    <e-column field='FirstName' label='First Name'
type='string' />
                    <e-column field='TitleOfCourtesy' label='Title Of
Courtesy' type='boolean' :values="values"/>
                    <e-column field='Title' label='Title' type='string' />
                    <e-column field='HireDate' label='Hire Date' type='date'
format='dd/MM/yyyy' />
                    <e-column field='Country' label='Country' type='string'
/>
                    <e-column field='City' label='City' type='string' />
                </e-columns>
            </ejs-querybuilder>
             <ejs-button cssClass="e-qb-button" :isPrimary="true" v-
on:click.native="addRule">Add Rule</ejs-button>
             <ejs-button cssClass="e-qbr-button" :isPrimary="true" v-
on:click.native="addGroup">Add Group</ejs-button>
             <ejs-button cssClass="e-qbr-button" :isPrimary="true" v-
on:click.native="deleteGroup">Delete Group</ejs-button>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(ButtonPlugin);
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: employeeData,
            values: ['Mr.', 'Mrs.'],
            showButtons:{ruleDelete: true, groupInsert: true, groupDelete:
true},
            importRules: {
                'condition': 'and',
                'rules': [{
                    'label': 'Employee ID',
                    'field': 'EmployeeID',
                    'type': 'number',
                    'operator': 'equal',
                    'value': 1
                },
                {
                    'label': 'Title',
                    'field': 'Title',
                    'type': 'string',
                    'operator': 'equal',
                    'value': 'Sales Manager'
                }]
            }
        };
    },
     methods: {
    addRule: function(event) {
       this.$refs.querybuilder.ej2Instances.addRules([{'label':
'City','field': 'City','type': 'string','operator': 'equal','value': 'US'}],
'group0');
    }
     addGroup: function(event) {
        this.$refs.querybuilder.ej2Instances.addGroups([{'condition':
'and','rules': [{'label': 'First Name','field': 'FirstName','type':
'string','operator': 'startswith','value': 'v' }]}], 'group0');
    }
     deleteGroup: function(event) {
        this.$refs.querybuilder.ej2Instances.deleteGroups(['group1']);
    }
  }
}
var employeeData = [{
      'EmployeeID': 1,
      'FirstName': 'Nancy',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Seattle',
      'Country': 'USA'
    },
    {
      'EmployeeID': 2,
      'FirstName': 'Andrew',
      'Title': 'Vice President',
```

```
      'TitleOfCourtesy': 'Dr.',
      'HireDate': '21/04/2003',
      'City': 'Tacoma',
      'Country': 'USA'
    },
    {
      'EmployeeID': 3,
      'FirstName': 'Janet',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Kirkland',
      'Country': 'USA'
    }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
    .e-qb-button {
  margin: 2% 1% 0 15%;
}
.e-qbr-button {
  margin: 2% 1% 0 1%;
}
#datatable {
  margin-left: 15%;
  margin-top: 2%;
}
#datatable.e-table {
  border: solid 1px #e0e0e0;
  border-collapse: collapse;
  font-family: Roboto;
}
#datatable.e-table td, #datatable.e-table th {
  border: solid #e0e0e0;
  border-width: 1px 0 0;
  display: table-cell;
  font-size: 14px;
  line-height: 20px;
  overflow: hidden;
  padding: 8px 21px;
  vertical-align: middle;
  white-space: nowrap;
  width: auto;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs9" %}

Templates in Vue Query builder component

Templates allows users to define customized header and own user interface for columns.

Header Template

Header Template allows to define your own user interface for Header, which includes creating or deleting rules and groups and to customize the AND/OR condition and NOT condition options. To implement header template, you can create the user interface as vue component and assign the values when requestType is header-template-create in actionBegin event.

In the following sample dropdown, splitbutton and button are used as the custom components in the header.

APP.VUE

```
<template>
    <div class="control-section">
            <ejs-querybuilder id="querybuilder" ref="querybuilder"
:dataSource="dataSource" :rule="importRules"
:headerTemplate="headerTemplate" enableNotCondition = true>
                <e-columns>
                    <e-column field="EmployeeID" label="Employee ID"
type="number"/>
                    <e-column field="FirstName" label="First Name"
type="string"/>
                    <e-column field="Age" label="Age" type="number" />
                    <e-column field='City' label='City' type='string' />
                    <e-column field='Country' label='Country' type='string'
/>
                </e-columns>
            </ejs-querybuilder>
            </div>
  </template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from '@syncfusion/ej2-vue-querybuilder';
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { getComponent, compile,closest } from '@syncfusion/ej2-base';
import { DataManager, Predicate, Query } from '@syncfusion/ej2-data';
import { CheckBoxPlugin, ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { DropDownButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
Vue.use(QueryBuilderPlugin);
Vue.use(SliderPlugin);
Vue.use(DropDownListPlugin);
Vue.use(CheckBoxPlugin);
Vue.use(DropDownButtonPlugin);
Vue.use(ButtonPlugin);
export default {
    data: function() {
        return {
            dataSource: employeeData,
            importRules: {
```

```
                'condition': 'and', 'not': true,
                'rules': [{
                    'label': 'Age',
                    'field': 'Age',
                    'type': 'number',
                    'operator': 'equal',
                    'value': 34
                },
                {
                    'label': 'FirstName',
                    'field': 'FirstName',
                    'type': 'string',
                    'operator': 'equal',
                    'value': 'Nancy'
                },
                {
                    'condition': 'or',
                    'rules': [{
                        'label': 'Age',
                        'field': 'Age',
                        'type': 'number',
                        'operator': 'equal',
                        'value': 34
                    }]
                }]
            },
            headerTemplate: () => {
                return {
                    template : Vue.component('headerTemplate', {
                        template:
                        `<div class="e-groupheader">
                            <button  v-if="data.notCondition !== undefined"
class='e-cb-wrapper'>
                             <ejs-checkbox :id="notID" label='not'
:checked="data.notCondition" :change="onChange"></ejs-checkbox>
                            </button>
                            <ejs-dropdownlist :id="ddlID"  cssClass="e-
custom-group-btn"  :dataSource="ds" :fields="fields"  v-
model="data.condition"  :change="conditionChange"></ejs-dropdownlist>
                            <ejs-dropdownbutton :id = 'ddbID'
:items='ddbitems' cssClass= "e-round e-small e-caret-hide e-addrulegroup e-
add-btn" iconCss="e-icons e-add-icon" :select='onSelect'></ejs-
dropdownbutton>
                            <ejs-button v-if="data.ruleID !==
'querybuilder_group0'" :id="dltbtnID" class= "e-btn e-delete-btn e-lib e-
small e-round e-icon-btn" iconCss="e-btn-icon e-icons e-delete-icon" v-
on:click.native='btnClick'>
                            </ejs-button>
                        </div>`,
                        data(args) {
                            return {
                                qryBldrObj:
getComponent(document.getElementById('querybuilder'), 'query-builder'),
                                ds:  [{'key': 'AND', 'value': 'and'},{'key':
'OR', 'value': 'or'}],
                                fields: { text: 'key', value: 'value' },
                                ddbitems:[
```

```
                                    {
                                        text: 'AddGroup',
                                        iconCss: 'e-icons e-add-icon e-
addgroup'
                                    },
                                    {
                                        text: 'AddCondition',
                                        iconCss: 'e-icons e-add-icon e-
addrule'
                                    }
                                ]
                            }
                        },
                        computed: {
                            ddbID: function(){
                                return `${this.data.ruleID}_addbtn`;
                            },
                            notID: function(){
                                return `${this.data.ruleID}_notOption`;
                            },
                            ddlID: function(){
                                return `${this.data.ruleID}_cndtn`;
                            },
                            dltbtnID: function(){
                                return `${this.data.ruleID}_dltbtn`;
                            }
                        },
                        methods: {
                            onSelect: function(event){
                               var addbtn = closest(event.element,'.e-
dropdown-popup');
                               var ddbId = addbtn.id; var ddb =
ddbId.split('_');
                               if (event.item.text === 'AddGroup') {
                                   this.qryBldrObj.addGroups([{condition:
'or', 'rules': [{}], not: false}], ddb[1]);
                                } else if (event.item.text ===
'AddCondition') {
                                    this.qryBldrObj.addRules([{}], ddb[1]);
                                }
                            },
                            onChange: function(args){
                                this.qryBldrObj.notifyChange(args.checked,
args.event.target, 'not');
                            },
                            conditionChange: function(args){
                                this.qryBldrObj.notifyChange(args.value,
args.element, 'condition');
                            },
                            btnClick: function(args){

this.qryBldrObj.deleteGroup(closest(args.target.offsetParent, '.e-group-
container'));
                            }
                        }
                    })
                }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            }
        };
    }
}
var employeeData = [
    { 'EmployeeID': 1, 'FirstName': 'Nancy', 'Age': 31, 'City': 'Seattle',
'Country': 'USA' },
    { 'EmployeeID': 2, 'FirstName': 'Andrew', 'Age': 32, 'City': 'Tacoma',
'Country': 'USA' },
    { 'EmployeeID': 3, 'FirstName': 'Janet', 'Age': 33, 'City': 'Kirkland',
'Country': 'USA' },
    { 'EmployeeID': 4, 'FirstName': 'Margaret', 'Age': 33, 'City':
'Redmond', 'Country': 'USA' },
    { 'EmployeeID': 5, 'FirstName': 'Steven', 'Age': 34, 'City': 'London',
'Country': 'UK' },
    { 'EmployeeID': 6, 'FirstName': 'Michael', 'Age': 35, 'City': 'London',
'Country': 'UK' },
    { 'EmployeeID': 7, 'FirstName': 'Robert', 'Age': 36, 'City': 'London',
'Country': 'UK' },
    { 'EmployeeID': 8, 'FirstName': 'Laura', 'Age': 37, 'City': 'Seattle',
'Country': 'USA' },
    { 'EmployeeID': 9, 'FirstName': 'Anne', 'Age': 38, 'City': 'London',
'Country': 'UK' }
]
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
.e-query-builder {
    margin: 0 auto;
}
.e-rule-template {
    padding-bottom: 12px;
}
.e-query-builder .e-add-btn {
    margin-left: 10px;
    margin-right: 10px;
}
.e-query-builder .cndtnbtn.e-control.e-dropdownlist.e-lib.e-input {
    padding-left: 10px;
}
.e-query-builder span.e-custom-group-btn {
    max-width: 100px;
    border-radius: 5px !important;
    border-width: 1px !important;
}
.e-query-builder .e-custom-group-btn.e-input-focus::before, .e-custom-group-
btn.e-input-focus::after {
```

```
    background: transparent !important;
}
.e-query-builder .e-group-header .e-addrulegroup, .e-group-header .e-delete-
btn {
    border: 1px solid grey !important;
}
.e-query-builder .e-group-header .e-addrulegroup:hover, .e-group-header .e-
delete-btn:hover {
    border: 1px solid grey !important;
}
.e-query-builder .e-toggle{
    background: #317ab9;
    border-color: #317ab9;
    color: #fff;
}
.e-query-builder .e-cb-wrapper {
    margin-right: 5px;
    height: 32px;
    border-radius: 5px;
    border: 1px solid gray;
    background-color: white;
}
.e-query-builder .e-custom-group-btn{
    padding-left: 10px;
    height: 32px;
}
.e-control {
display: inline;
}
.e-query-builder div.e-rule-container{
   width: 800px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/header-template-cs1" %}

Column Template

Template allows you to define your own input widgets for columns. To implement template, you can define the following functions

- create: Creates the custom component.
- write: Wire events for the custom component.
- Destroy: Destroy the custom component.

In the following sample, dropdown is used as the custom component in the PaymentMode column.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder ref="querybuilder" :dataSource="dataSource"
:rule="importRules" width="70%">
                <e-columns>
```

```
                    <e-column field='Category' label='Category'
type='string' />
                    <e-column field='PaymentMode' label='Payment Mode'
type='string' :template='paymentTemplate' />
                    <e-column field='TransactionType' label='Transaction
Type' type='boolean' />
                    <e-column field='Description' label='Description'
type='string' />
                    <e-column field='Date' label='Date' type='date' />
                    <e-column field='Amount' label='Amount' type='number' />
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { DropDownList } from '@syncfusion/ej2-dropdowns'
import { createElement, getComponent} from "@syncfusion/ej2-base";
let inOperators = ['in', 'notin'];
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: expenseData,
             paymentTemplate: {
                create: () => {
                    return createElement('input', { attrs: { 'type': 'text'
} });
                },
                destroy: (args) => {
                    let multiselect =
getComponent(document.getElementById(args.elementId), 'multiselect');
                    if(multiselect)
                        multiselect.destroy();
                    let dropdownlist =
getComponent(document.getElementById(args.elementId), 'dropdownlist');
                    if(dropdownlist)
                        dropdownlist.destroy();
                },
                write: (args) => {
                    let ds = ['Cash', 'Debit Card', 'Credit Card', 'Net
Banking', 'Wallet'];
                    if (inOperators.indexOf(args.operator) > -1) {
                        let multiSelectObj = new MultiSelect({
                            dataSource: ds,
                            value: args.values,
                            mode: 'CheckBox',
                            placeholder: 'Select Transaction',
                            change: (e) => {

this.$refs.querybuilder.$el.ej2_instances[0].notifyChange(e.value,
e.element);
                            }
                        });
                        multiSelectObj.appendTo('#' + args.elements.id);
```

```
                    } else {
                        let dropDownObj = new DropDownList({
                            dataSource: ds,
                            value: args.values,
                            change: (e) => {

this.$refs.querybuilder.$el.ej2_instances[0].notifyChange(e.itemData.value,
e.element);
                            }
                        });
                        dropDownObj.appendTo('#' + args.elements.id);
                    }
                }
            },
            importRules: {
        'condition': 'and',
        'rules': [{
                'label': 'Payment Mode',
                'field': 'PaymentMode',
                'type': 'string',
                'operator': 'equal',
                'value': 'Cash'
            }]
        }
        };
    }
}
var expenseData = [{
    'Category': 'Food',
    'PaymentMode': 'Credit Card',
    'TransactionType': 'Expense',
    'Description': 'Boiled peanuts',
    'Amount': '7',
    'Date': '06/01/2017'
  }, {
    'Category': 'Food',
    'PaymentMode': 'Cash',
    'TransactionType': 'Expense',
    'Description': 'Peanuts in Coke',
    'Amount': '8',
    'Date': '06/04/2017'
  }, {
    'Category': 'Food',
    'PaymentMode': 'Cash',
    'TransactionType': 'Expense',
    'Description': 'Palmetto Cheese, Mint julep',
    'Amount': '11',
    'Date': '06/04/2017'
  }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs23" %}

Using Template

Template allows you to define your own input widgets for columns. To implement template, you can create the user interface as vue component.

APP.VUE

```
<template>
    <div class="control-section">
            <ejs-querybuilder  id="querybuilder" ref="querybuilder"
:rule="importRules" width="100%" >
                <e-columns>
                    <e-column field='Category' label='Category'
type='string'/>
                    <e-column field='PaymentMode' label='Payment Mode'
type='string' :operators="customOperators" :template='paymentTemplate' />
                    <e-column field='TransactionType' label='Transaction
Type' type='string' :operators="customOperators"
:template='transactionTemplate' />
                    <e-column field='Description' label='Description'
type='string' />
                    <e-column field='Date' label='Date' type='date' />
                    <e-column field='Amount' label='Amount' type='number' />
                </e-columns>
            </ejs-querybuilder>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { RadioButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { DropDownList, MultiSelect, CheckBoxSelection } from
'@syncfusion/ej2-dropdowns';
import { Slider, TextBox } from '@syncfusion/ej2-inputs';
import { CheckBox } from '@syncfusion/ej2-buttons';
import { createElement, getComponent, isNullOrUndefined } from
"@syncfusion/ej2-base";
import { CheckBoxPlugin } from "@syncfusion/ej2-vue-buttons";
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
MultiSelect.Inject(CheckBoxSelection);
Vue.use(QueryBuilderPlugin);
Vue.use(CheckBoxPlugin);
Vue.use(DropDownListPlugin);
export default {
    data: function() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        return {
            customOperators: [{value: 'equal', key: 'Equal'}, {value:
'notequal', key: 'Not Equal'}],
            importRules:{
                'condition': 'or',
                'rules': [{
                    'label': 'TransactionType',
                    'field': 'TransactionType',
                    'type': 'boolean',
                    'operator': 'equal',
                    'value': 'Income'
                },
                {
                    'label': 'PaymentMode',
                    'field': 'PaymentMode',
                    'type': 'string',
                    'operator': 'equal',
                    'value': 'Cash'
                }]
            },
            paymentTemplate: () => {
                return {
                    template: Vue.component('paymentTemplate', {
                        template:
                        `<div >
                        <ejs-dropdownlist  :dataSource="ds"  v-
model="data.rule.value" :change="paymentChange"></ejs-dropdownlist>
                        </div>`,
                        data(args) {
                            return{
                               qryBldrObj:
getComponent(document.getElementById('querybuilder'), 'query-builder'),
                                data: Object.assign({}, args, true),
                                ds: ['Cash', 'Debit Card', 'Credit Card',
'Net Banking'];
                            }
                        },
                        methods: {
                            paymentChange: function(event){
                                var elem =
document.getElementById(ruleID).querySelector('.e-rule-value');
                                this.qryBldrObj.notifyChange(event.value as
string, elem, 'value');
                            }
                        }
                    })
                }
            },
            transactionTemplate: () => {
                return{
                    template: Vue.component('transactionTemplate', {
                        template:
                        `<div >
                        <ejs-checkbox  label='Is Expense'
:checked="data.rule.value" :change="transactionChange"></ejs-checkbox>
                        </div>`,
                        data(args) {
```

```
                                return{
                                    qryBldrObj:
getComponent(document.getElementById('querybuilder'), 'query-builder'),
                                    data: Object.assign({}, args, true)
                                }
                            },
                            methods: {
                               transactionChange: function(event){
                                   var elem =
document.getElementById(ruleID).querySelector('.e-rule-value');
                                   this.qryBldrObj.notifyChange(event.checked
=== true ? 'Expense' : 'Income', elem, 'value');
                                }
                            }
                        })
                    }
                }
            };
        }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/query-builder/template-cs1" %}

Rule Template

Rule Template allows to define your own user interface for columns. To implement ruleTemplate, you can create the user interface as vue component and assign the values through actionBegin event.

In the following sample, dropdown and slider are used as the custom component and applied greaterthanorequal operator to Age column.

APP.VUE

```
<template>
    <div class="control-section">
        <div id="listbox-selection">
            <ejs-querybuilder id="querybuilder" ref="querybuilder"
:actionBegin="actionBegin" :dataSource="dataSource" :rule="importRules">
                <e-columns>
                    <e-column field="EmployeeID" label="Employee ID"
type="number"/>
                    <e-column field="FirstName" label="First Name"
type="string"/>
                    <e-column field="LastName" label="LastName"
type="string"/>
                    <e-column field="Age" label="Age" type="number"
:ruleTemplate='ageTemplate'/>
                    <e-column field='City' label='City' type='string' />
                    <e-column field='Country' label='Country' type='string'
/>
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { QueryBuilderPlugin } from '@syncfusion/ej2-vue-querybuilder';
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { getComponent, compile } from '@syncfusion/ej2-base';
import { DataManager, Predicate, Query } from '@syncfusion/ej2-data';
Vue.use(QueryBuilderPlugin);
Vue.use(SliderPlugin);
Vue.use(DropDownListPlugin);
export default {
    data: function() {
        return {
            dataSource: employeeData,
            importRules: {
                'condition': 'and',
                'rules': [{
                    'label': 'Age',
                    'field': 'Age',
                    'type': 'number',
                    'operator': 'greaterthanorequal',
                    'value': 30
                }]
            },
            ageTemplate: () => {
                return {
                    template : Vue.component('ruleTemplate', {
                        template:
                        `<div class="e-rule e-rule-template">
                            <div class="e-rule-filter e-custom-filter">
                                <ejs-dropdownlist :change='fieldChange'
:value="data.rule.field" :dataSource="data.columns" :fields="data.fields">
                                </ejs-dropdownlist>
                            </div>
                            <div>
                                <div class="e-slider-value">
                                    <ejs-slider :min="min" :max="max"
ref="slider" :ticks="rangeticks" :change="valueChange" :value="value"
:id="valueID">
                                    </ejs-slider>
                                </div>
                                <div class="e-rule-btn">
                                    <button class="e-removerule e-rule-
delete e-css e-btn e-small e-round">
                                        <span class="e-btn-icon e-icons e-
delete-icon"/>
                                    </button>
                                </div>
                            </div>
                        </div>`,
                        data(args) {
                            return {
                                qryBldrObj:
getComponent(document.getElementById('querybuilder'), 'query-builder'),
                                rangeticks: {
                                    placement: 'Before',
                                    largeStep: 5,
                                    smallStep: 1,
```

```
                                    showSmallTicks: true
                                },
                                min: 30,
                                max: 50
                            }
                        },
                        computed: {
                            valueID: function() {
                                return `${this.data.ruleID}_valuekey0`;
                            },
                            value: function() {
                                var num = 30;
                                if (this.data.rule.value !== '') {
                                    num = this.data.rule.value;
                                }
                                return num;
                            }
                        },
                        methods: {
                            fieldChange: function(args) {
                                this.qryBldrObj.notifyChange(args.value,
args.element, 'field');
                            },
                            valueChange: function(args) {
                                if (args.isInteracted) {
                                    var elem = this.$refs.slider.$el;
                                    this.qryBldrObj.notifyChange(args.value,
elem, 'value');
                                }
                            }
                        }
                    })
                }
            }
        };
    },
    methods: {
        actionBegin: function(args) {
            args.rule.operator = 'greaterthanorequal';
            if (args.requestType === 'template-initialize') {
                if (args.rule.value === '') {
                    args.rule.value = 30;
                }
            }
            if (args.requestType === 'template-create') {
                getComponent(document.getElementById(args.ruleID +
'_valuekey0'), 'slider').refresh();
            }
        }
    }
}
var employeeData = [
    { 'EmployeeID': 1, 'FirstName': 'Nancy', 'Age': 31, 'City': 'Seattle',
'Country': 'USA' },
    { 'EmployeeID': 2, 'FirstName': 'Andrew', 'Age': 32, 'City': 'Tacoma',
'Country': 'USA' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    { 'EmployeeID': 3, 'FirstName': 'Janet', 'Age': 33, 'City': 'Kirkland',
'Country': 'USA' },
    { 'EmployeeID': 4, 'FirstName': 'Margaret', 'Age': 33, 'City':
'Redmond', 'Country': 'USA' },
    { 'EmployeeID': 5, 'FirstName': 'Steven', 'Age': 34, 'City': 'London',
'Country': 'UK' },
    { 'EmployeeID': 6, 'FirstName': 'Michael', 'Age': 35, 'City': 'London',
'Country': 'UK' },
    { 'EmployeeID': 7, 'FirstName': 'Robert', 'Age': 36, 'City': 'London',
'Country': 'UK' },
    { 'EmployeeID': 8, 'FirstName': 'Laura', 'Age': 37, 'City': 'Seattle',
'Country': 'USA' },
    { 'EmployeeID': 9, 'FirstName': 'Anne', 'Age': 38, 'City': 'London',
'Country': 'UK' }
];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
    .e-rule-template {
        padding-bottom: 12px;
    }
    .e-rule-btn {
        float: right;
        padding-top: 12px;
    }
    .e-rule-value-group {
        margin: 12px;
        border-top: 1px solid #e0e0e0;
        min-height: 40px;
    }
    .e-query-builder .e-slider-container.e-horizontal {
        padding: 0px 0 0 18px;
        height: 0;
    }
    .e-query-builder .e-hide {
        display: none;
    }
    .e-query-builder .e-custom-filter {
        width: 40% !important;
    }
    .e-query-builder .e-slider-container.e-horizontal .e-slider {
        top: calc(50% - 10px);
    }
    .e-query-builder .e-slider-value {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        width: 40% !important;
        padding: 12px 0 12px 0;
        display: inline-block;
    }
    .e-rule-table {
        margin: 20px 0 0 20px;
        border: solid 1px #e0e0e0;
        border-collapse: collapse;
        font-family: Roboto;
        width: 320px;
        height: 180px;
        overflow-y: auto;
    }
    .e-rule-table th,
    .e-rule-table td {
        border: solid #e0e0e0;
        border-width: 1px 0 0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/rule-template-cs1" %}

Model binding in Vue Query builder component

Model binding allows to bind properties for the components used in field, operator, and value columns. To implement model binding, assign fieldModel, operatorModel, and valueModel properties in QueryBuilder.

APP.VUE

```
<template>
    <div class="control-section">
            <ejs-querybuilder id="querybuilder" ref="querybuilder"
:dataSource="dataSource" :rule="importRules" enableNotCondition = true
:fieldModel = "{allowFiltering: true, popupHeight: '500px'}" :operatorModel
= "{allowFiltering: true, popupHeight: '400px'}" :valueModel =
"{numericTextBoxModel:
        {
            cssClass: 'e-custom'
        },
        multiSelectModel: {
            cssClass: 'e-custom'
        },
        datePickerModel: {
            cssClass: 'e-custom'
        },
        textBoxModel: {
            cssClass: 'e-custom'
        },
```

```
        radioButtonModel: {
            cssClass: 'e-custom'
        }}">
                <e-columns>
                    <e-column field="EmployeeID" label="Employee ID"
type="number"/>
                    <e-column field="FirstName" label="First Name"
type="string"/>
                    <e-column field="Age" label="Age" type="number" />
                    <e-column field='City' label='City' type='string' />
                    <e-column field='Country' label='Country' type='string'
/>
                </e-columns>
            </ejs-querybuilder>
            </div>
  </template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from '@syncfusion/ej2-vue-querybuilder';
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { getComponent, compile,closest } from '@syncfusion/ej2-base';
import { DataManager, Predicate, Query } from '@syncfusion/ej2-data';
import { CheckBoxPlugin, ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { DropDownButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
Vue.use(QueryBuilderPlugin);
Vue.use(SliderPlugin);
Vue.use(DropDownListPlugin);
Vue.use(CheckBoxPlugin);
Vue.use(DropDownButtonPlugin);
Vue.use(ButtonPlugin);
export default {
    data: function() {
        return {
            dataSource: employeeData,
            importRules: {
                'condition': 'and',
                'rules': [{
                    'label': 'Employee ID',
                    'field': 'EmployeeID',
                    'type': 'number',
                    'operator': 'equal',
                    'value': 1001
                }]
            }
        };
    }
}
var employeeData = [
    { 'EmployeeID': 1, 'FirstName': 'Nancy', 'Age': 31, 'City': 'Seattle',
'Country': 'USA' },
    { 'EmployeeID': 2, 'FirstName': 'Andrew', 'Age': 32, 'City': 'Tacoma',
'Country': 'USA' },
    { 'EmployeeID': 3, 'FirstName': 'Janet', 'Age': 33, 'City': 'Kirkland',
'Country': 'USA' },
    { 'EmployeeID': 4, 'FirstName': 'Margaret', 'Age': 33, 'City':
'Redmond', 'Country': 'USA' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    { 'EmployeeID': 5, 'FirstName': 'Steven', 'Age': 34, 'City': 'London',
'Country': 'UK' },
    { 'EmployeeID': 6, 'FirstName': 'Michael', 'Age': 35, 'City': 'London',
'Country': 'UK' },
    { 'EmployeeID': 7, 'FirstName': 'Robert', 'Age': 36, 'City': 'London',
'Country': 'UK' },
    { 'EmployeeID': 8, 'FirstName': 'Laura', 'Age': 37, 'City': 'Seattle',
'Country': 'USA' },
    { 'EmployeeID': 9, 'FirstName': 'Anne', 'Age': 38, 'City': 'London',
'Country': 'UK' }
]
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/model-binding-cs1" %}

Import export in Vue Query builder component

Importing allows you to view or edit the predefined conditions which is available in JSON or SQL. You can import the conditions either initial rendering or post rendering.

Initial rendering

To apply conditions initially, you can define the rule Here, you can import structured JSON object by defining the rule property.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="dataSource"
:rule="importRules">
                <e-columns>
                  <e-column field="TaskID" label="Task ID"
type="number"></e-column>
                  <e-column field="Name" label="Name" type="string"></e-
column>
                  <e-column field="Category" label="Category"
type="string"></e-column>
                  <e-column field="SerialNo" label="SerialNo"
type="string"></e-column>
                  <e-column field="InvoiceNo" label="InvoiceNo"
type="string"></e-column>
                  <e-column field="Status" label="Status" type="string"></e-
column>
```

```
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: hardwareData,
            importRules:  {
        'condition': 'or',
        'rules': [{
            'label': 'Category',
            'field': 'Category',
            'type': 'string',
            'operator': 'equal',
            'value': 'Laptop'
        },
        {
        'condition': 'and',
            'rules': [{
                'label': 'Status',
                'field': 'Status',
                'type': 'string',
                'operator': 'notequal',
                'value': 'Pending'
            },
            {
                'label': 'Task ID',
                'field': 'TaskID',
                'type': 'number',
                'operator': 'equal',
                'value': 5675
            }]
        }]
    }
        };
    }
}
var hardwareData = [{
      'TaskID': 1,
      'Name': 'Lenovo Yoga',
      'Category': 'Laptop',
      'SerialNo': 'CB27932009',
      'InvoiceNo': 'INV-2878',
      'Status': 'Assigned'
  },
  {
      'TaskID': 2,
      'Name': 'Acer Aspire',
      'Category': 'Others',
      'SerialNo': 'CB35728290',
      'InvoiceNo': 'INV-3456',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        'Status': 'In-repair'
    },
    {
        'TaskID': 3,
        'Name': 'Apple MacBook',
        'Category': 'Laptop',
        'SerialNo': 'CB35628728',
        'InvoiceNo': 'INV-2763',
        'Status': 'In-repair'
    }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs19" %}

Post rendering

Importing from JSON

You can set the conditions from structured JSON object through the setRules method.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" ref="querybuilder"
:dataSource="dataSource" >
                <e-columns>
                  <e-column field="TaskID" label="Task ID"
type="number"></e-column>
                  <e-column field="Name" label="Name" type="string"></e-
column>
                  <e-column field="Category" label="Category"
type="string"></e-column>
                  <e-column field="SerialNo" label="SerialNo"
type="string"></e-column>
                  <e-column field="InvoiceNo" label="InvoiceNo"
type="string"></e-column>
                  <e-column field="Status" label="Status" type="string"></e-
column>
                </e-columns>
            </ejs-querybuilder>
```

```
                <ejs-button cssClass="e-qb-button" :isPrimary="true" v-
on:click.native="btnClick">Set Rules</ejs-button>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: hardwareData,
            importRules:  {
        'condition': 'or',
        'rules': [{
            'label': 'Category',
            'field': 'Category',
            'type': 'string',
            'operator': 'equal',
            'value': 'Laptop'
        },
        {
        'condition': 'and',
            'rules': [{
                'label': 'Status',
                'field': 'Status',
                'type': 'string',
                'operator': 'notequal',
                'value': 'Pending'
            },
            {
                'label': 'Task ID',
                'field': 'TaskID',
                'type': 'number',
                'operator': 'equal',
                'value': 5675
            }]
        }]
    }
        };
    },
      methods: {
    btnClick: function(event) {
      this.$refs.querybuilder.ej2Instances.setRules(this.importRules);
    }
  }
}
var hardwareData = [{
      'TaskID': 1,
      'Name': 'Lenovo Yoga',
      'Category': 'Laptop',
      'SerialNo': 'CB27932009',
      'InvoiceNo': 'INV-2878',
      'Status': 'Assigned'
```

```
  },
  {
      'TaskID': 2,
      'Name': 'Acer Aspire',
      'Category': 'Others',
      'SerialNo': 'CB35728290',
      'InvoiceNo': 'INV-3456',
      'Status': 'In-repair'
  },
  {
      'TaskID': 3,
      'Name': 'Apple MacBook',
      'Category': 'Laptop',
      'SerialNo': 'CB35628728',
      'InvoiceNo': 'INV-2763',
      'Status': 'In-repair'
  }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
     .e-qb-button {
  margin: 2% 1% 0 15%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs20" %}

Importing from SQL

You can set the conditions from SQL query through the setRulesFromSql method.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" ref="querybuilder"
:dataSource="dataSource" :rule="importRules">
                <e-columns>
                  <e-column field="TaskID" label="Task ID"
type="number"></e-column>
                  <e-column field="Name" label="Name" type="string"></e-
column>
```

```
                    <e-column field="Category" label="Category"
type="string"></e-column>
                    <e-column field="SerialNo" label="SerialNo"
type="string"></e-column>
                    <e-column field="InvoiceNo" label="InvoiceNo"
type="string"></e-column>
                    <e-column field="Status" label="Status" type="string"></e-
column>
                </e-columns>
            </ejs-querybuilder>
            <ejs-button cssClass="e-qb-button" :isPrimary="true" v-
on:click.native="btnClick">Set Rules</ejs-button>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: hardwareData,
            importRules: {
        'condition': 'or',
        'rules': [{
            'label': 'Category',
            'field': 'Category',
            'type': 'string',
            'operator': 'equal',
            'value': 'Laptop'
        }]
    }
        };
    },
    methods: {
    btnClick: function(event) {
     this.$refs.querybuilder.ej2Instances.setRulesFromSql("TaskID = 1 and
Status LIKE ('Assigned%')");
    }
  }
}
var hardwareData = [{
      'TaskID': 1,
      'Name': 'Lenovo Yoga',
      'Category': 'Laptop',
      'SerialNo': 'CB27932009',
      'InvoiceNo': 'INV-2878',
      'Status': 'Assigned'
  },
  {
      'TaskID': 2,
      'Name': 'Acer Aspire',
      'Category': 'Others',
      'SerialNo': 'CB35728290',
```

```
        'InvoiceNo': 'INV-3456',
        'Status': 'In-repair'
    },
    {
        'TaskID': 3,
        'Name': 'Apple MacBook',
        'Category': 'Laptop',
        'SerialNo': 'CB35628728',
        'InvoiceNo': 'INV-2763',
        'Status': 'In-repair'
    }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
     .e-qb-button {
     margin: 2% 1% 0 15%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs21" %}

Exporting

Exporting allows you to save or maintain the created conditions through the Query Builder. You can export the defined conditions by the following ways.

Exporting to JSON

You can export the defined conditions to structured JSON object through the getRules method.

Exporting to SQL

You can export the defined conditions to SQL query through the getSqlFromRules method.

Install Syncfusion Popup packages using below command.

`bash

npm install @syncfusion/ej2-vue-popups --save

`

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
```

```
            <ejs-querybuilder width="70%"  ref="querybuilder"
:dataSource="dataSource" :rule="importRules">
                <e-columns>
                  <e-column field="TaskID" label="Task ID"
type="number"></e-column>
                  <e-column field="Name" label="Name" type="string"></e-
column>
                  <e-column field="Category" label="Category"
type="string"></e-column>
                  <e-column field="SerialNo" label="SerialNo"
type="string"></e-column>
                  <e-column field="InvoiceNo" label="InvoiceNo"
type="string"></e-column>
                  <e-column field="Status" label="Status" type="string"></e-
column>
                </e-columns>
            </ejs-querybuilder>
            <ejs-button cssClass="e-qb-button" :isPrimary="true" v-
on:click.native="getSql">Get SQL</ejs-button>
            <ejs-button cssClass="e-qbr-button" :isPrimary="true" v-
on:click.native="getJson">Get JSON</ejs-button>
            <ejs-dialog ref="dialogObj" :header="header"
:animationSettings="animationSettings" :showCloseIcon="true"
:visible="false" width="500px" height="80%">
            </ejs-dialog>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { DialogPlugin } from '@syncfusion/ej2-vue-popups';
Vue.use(DialogPlugin);
Vue.use(ButtonPlugin);
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: hardwareData,
            importRules: {
        'condition': 'or',
        'rules': [{
            'label': 'Category',
            'field': 'Category',
            'type': 'string',
            'operator': 'equal',
            'value': 'Laptop'
        }]
    },
    header: 'Querybuilder rule',
      animationSettings: { effect: 'Zoom', duration: 400 },
        };
    },
    methods: {
    getSql: function(event) {
```

```
            this.$refs.dialogObj.content =
this.$refs.querybuilder.ej2Instances.getSqlFromRules(this.$refs.querybuilder
.ej2Instances.getRules());
            this.$refs.dialogObj.show();
        }
        getJson: function(event) {
            var validRule =
this.$refs.querybuilder.ej2Instances.getValidRules(this.$refs.querybuilder.e
j2Instances.rule);
            this.$refs.dialogObj.content =  '<pre>' + JSON.stringify(validRule,
null, 4) + '</pre>';
            this.$refs.dialogObj.show();
        }
    }
}
var hardwareData = [{
      'TaskID': 1,
      'Name': 'Lenovo Yoga',
      'Category': 'Laptop',
      'SerialNo': 'CB27932009',
      'InvoiceNo': 'INV-2878',
      'Status': 'Assigned'
  },
  {
      'TaskID': 2,
      'Name': 'Acer Aspire',
      'Category': 'Others',
      'SerialNo': 'CB35728290',
      'InvoiceNo': 'INV-3456',
      'Status': 'In-repair'
  },
  {
      'TaskID': 3,
      'Name': 'Apple MacBook',
      'Category': 'Laptop',
      'SerialNo': 'CB35628728',
      'InvoiceNo': 'INV-2763',
      'Status': 'In-repair'
  }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
    .e-qb-button {
       margin: 2% 1% 0 15%;
```

```
    }
    .e-qbr-button {
        margin: 2% 1% 0 1%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs22" %}

Global local in Vue Query builder component

The Localization library allows you to localize default text content of the Query Builder. The Query Builder component has static text that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the locale value and translation object.

The following list of properties and its values are used in the Query Builder.

| Locale key words | Text |

| ---------- | ---------- |

| AddGroup | Add Group |

| AddCondition | Add Condition |

| DeleteRule | Remove this condition |

| DeleteGroup | Delete group |

| Edit | EDIT |

| SelectField | Select a field |

| SelectOperator | Select operator |

| StartsWith | Starts With|

| EndsWith | Ends With |

| Contains | Contains |

| Equal | Equal |

| NotEqual | Not Equal |

| LessThan | Less Than |

| LessThanOrEqual | Less Than Or Equal |

| GreaterThan | Greater Than |

| GreaterThanOrEqual | Greater Than Or Equal |

| Between | Between |

| NotBetween | Not Between|

| In | In |

| NotIn | Not In |

| Remove | REMOVE |

| ValidationMessage | This field is required |

Copyright © 2001 -2024 Syncfusion Inc.

| SummaryViewTitle | Summary View |

| OtherFields | Other Fields |

| AND | AND |

| OR | OR |

| SelectValue | Enter Value |

| IsEmpty | Is Empty |

| IsNotEmpty | Is Not Empty |

| IsNull | Is Null |

| IsNotNull | Is Not Null |

| True | True |

| False | False |

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="dataSource"
:rule="importRules" locale="de-DE">
                <e-columns>
                  <e-column field="TaskID" label="Task ID"
type="number"></e-column>
                  <e-column field="Name" label="Name" type="string"></e-
column>
                  <e-column field="Category" label="Category"
type="string"></e-column>
                  <e-column field="SerialNo" label="SerialNo"
type="string"></e-column>
                  <e-column field="InvoiceNo" label="InvoiceNo"
type="string"></e-column>
                  <e-column field="Status" label="Status" type="string"></e-
column>
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { L10n, setCulture } from '@syncfusion/ej2-base';
Vue.use(QueryBuilderPlugin);
setCulture('de-DE');
L10n.load({
    'de-DE': {
        'querybuilder': {
            'AddGroup': 'Gruppe hinzufügen',
            'AddCondition': 'Bedingung hinzufügen',
            'DeleteRule': 'Entfernen Sie diesen Zustand',
            'DeleteGroup': 'Gruppe löschen',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            'Edit': 'BEARBEITEN',
            'SelectField': 'Wählen Sie ein Feld aus',
            'DeleteRule': 'Entfernen Sie diesen Zustand',
            'DeleteGroup': 'Gruppe löschen',
            'SelectOperator': 'Operator auswählen',
            'StartsWith': 'Beginnt mit',
            'EndsWith': 'Endet mit',
            'Contains': 'Enthält',
            'Equal': 'Gleich',
            'NotEqual': 'Nicht gleich',
            'LessThan': 'Weniger als',
            'LessThanOrEqual': 'Weniger als oder gleich',
            'GreaterThan': 'Größer als',
            'GreaterThanOrEqual': 'Größer als oder gleich',
            'Between': 'Zwischen',
            'NotBetween': 'Nicht zwischen',
            'In': 'Im',
            'NotIn': 'Nicht in',
            'Remove': 'LÖSCHEN',
            'ValidationMessage': 'Dieses Feld wird benötigt',
            'True': 'Wahr',
            'False': 'Falsch',
        }
    }
});
export default {
    data: function() {
        return {
            dataSource: hardwareData,
            importRules: {
        'condition': 'or',
        'rules': [{
            'label': 'Category',
            'field': 'Category',
            'type': 'string',
            'operator': 'equal',
            'value': 'Laptop'
        }]
    }
        };
    }
}
var hardwareData = [{
      'TaskID': 1,
      'Name': 'Lenovo Yoga',
      'Category': 'Laptop',
      'SerialNo': 'CB27932009',
      'InvoiceNo': 'INV-2878',
      'Status': 'Assigned'
  },
  {
      'TaskID': 2,
      'Name': 'Acer Aspire',
      'Category': 'Others',
      'SerialNo': 'CB35728290',
      'InvoiceNo': 'INV-3456',
      'Status': 'In-repair'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
  {
      'TaskID': 3,
      'Name': 'Apple MacBook',
      'Category': 'Laptop',
      'SerialNo': 'CB35628728',
      'InvoiceNo': 'INV-2763',
      'Status': 'In-repair'
  }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs12" %}

Style and appearance in Vue Query builder component

To modify the QueryBuilder appearance, you need to override the default CSS of QueryBuilder component. Please find the list of CSS classes and its corresponding section in QueryBuilder component. Also, you have an option to create your own custom theme for the controls using our Theme Studio.

CSS Class | Purpose of Class

|.e-group-header .e-btn|To customize the condition button in querybuilder

|.e-group-body .e-rule-container|To customize the querybuilder rule container

|.e-group-container .e-group-header .e-dropdown-btn|To customize the querybuilder Add group/condition button

|.e-query-builder .e-group-header .e-deletegroup|To customize the querybuilder Delete group button

|.e-query-builder .e-rule-field .e-rule-value-delete .e-rule-delete|To customize the querybuilder Delete condition button

|.e-query-builder .e-rule-list > ::after,.e-query-builder .e-rule-list > ::before|To customize the querybuilder group joining line

|.e-query-builder .e-rule-container.e-joined-rule|To customize the querybuilder condition joining line

Accessibility in Vue Query Builder component

The Query Builder component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

Copyright © 2001 -2024 Syncfusion Inc.

The accessibility compliance for the Query Builder component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

WAI-ARIA (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components.

Copyright © 2001 -2024 Syncfusion Inc.

The following list of ARIA attributes is used in Query Builder.

| Attributes | Purpose |

| --- | --- |

| role | Indicates the query builder component. |

Keyboard interaction

The Query Builder component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Query Builder component.

| **Press** | **To do this** |

| --- | --- |

| Tab / Shift + Tab | To focus the next item in the rule. |

Ensuring accessibility

The Query Builder component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Query Builder component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Query Builder component with accessibility tools.

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs1" %}

See also

- Accessibility in Syncfusion Vue components

How To

Display mode in Vue Query builder component

Display options allow you to view the Query Builder in Vertically or Horizontally. For this, you should use the displayMode property.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="40%" :dataSource="dataSource"
:rule="importRules" displayMode="Vertical">
                <e-columns>
                    <e-column field='EmployeeID' label='Employee ID'
type='number' />
                    <e-column field='FirstName' label='First Name'
type='string' />
                    <e-column field='TitleOfCourtesy' label='Title Of
Courtesy' type='boolean' :values="values"/>
                    <e-column field='Title' label='Title' type='string' />
                    <e-column field='HireDate' label='Hire Date' type='date'
format='dd/MM/yyyy' />
                    <e-column field='Country' label='Country' type='string'
/>
```

```
                    <e-column field='City' label='City' type='string' />
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: employeeData,
            values: ['Mr.', 'Mrs.'],
            importRules: {
                'condition': 'and',
                'rules': [{
                    'label': 'Employee ID',
                    'field': 'EmployeeID',
                    'type': 'number',
                    'operator': 'equal',
                    'value': 1
                },
                {
                    'label': 'Title',
                    'field': 'Title',
                    'type': 'string',
                    'operator': 'equal',
                    'value': 'Sales Manager'
                }]
            }
        };
    }
}
var employeeData = [{
      'EmployeeID': 1,
      'FirstName': 'Nancy',
      'Title': 'Sales Representative',
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Seattle',
      'Country': 'USA'
    },
    {
      'EmployeeID': 2,
      'FirstName': 'Andrew',
      'Title': 'Vice President',
      'TitleOfCourtesy': 'Dr.',
      'HireDate': '21/04/2003',
      'City': 'Tacoma',
      'Country': 'USA'
    },
    {
      'EmployeeID': 3,
      'FirstName': 'Janet',
      'Title': 'Sales Representative',
```

```
      'TitleOfCourtesy': 'Ms.',
      'HireDate': '22/07/2001',
      'City': 'Kirkland',
      'Country': 'USA'
    }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs13" %}

The default view the query builder component is Horizontal.

The default view the query builder component in Vertical.

Sort columns in Vue Query builder component

SortDirection allows you to sort the columns bounded to the Query Builder to view the columns by ascending or descending order. You should set the sortDirection property to sort the fields.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="dataSource"
:rule="importRules" sortDirection="Ascending">
                <e-columns>
                  <e-column field="TaskID" label="Task ID"
type="number"></e-column>
                  <e-column field="Name" label="Name" type="string"></e-
column>
                  <e-column field="Category" label="Category"
type="string"></e-column>
                  <e-column field="SerialNo" label="SerialNo"
type="string"></e-column>
                  <e-column field="InvoiceNo" label="InvoiceNo"
type="string"></e-column>
                  <e-column field="Status" label="Status" type="string"></e-
column>
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: hardwareData,
            importRules: {
        'condition': 'or',
        'rules': [{
            'label': 'Category',
            'field': 'Category',
            'type': 'string',
            'operator': 'equal',
            'value': 'Laptop'
        },
        {
        'condition': 'and',
            'rules': [{
                'label': 'Status',
                'field': 'Status',
                'type': 'string',
                'operator': 'notequal',
                'value': 'Pending'
            },
            {
                'label': 'Task ID',
                'field': 'TaskID',
                'type': 'number',
                'operator': 'equal',
                'value': 5675
            }]
        }]
    }
        };
    }
}
var hardwareData = [{
      'TaskID': 1,
      'Name': 'Lenovo Yoga',
      'Category': 'Laptop',
      'SerialNo': 'CB27932009',
      'InvoiceNo': 'INV-2878',
      'Status': 'Assigned'
  },
  {
      'TaskID': 2,
      'Name': 'Acer Aspire',
      'Category': 'Others',
      'SerialNo': 'CB35728290',
      'InvoiceNo': 'INV-3456',
      'Status': 'In-repair'
  },
  {
      'TaskID': 3,
```

```
      'Name': 'Apple MacBook',
      'Category': 'Laptop',
      'SerialNo': 'CB35628728',
      'InvoiceNo': 'INV-2763',
      'Status': 'In-repair'
  }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs16" %}

Restrict groups in Vue Query builder component

You can restrict the condition set by defining the maxGroupCount property. By default, the value is 5. In the below demo, the maxGroupCount is set to 2 .

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="dataSource"
:rule="importRules"  maxGroupCount=2>
                <e-columns>
                  <e-column field="TaskID" label="Task ID"
type="number"></e-column>
                  <e-column field="Name" label="Name" type="string"></e-
column>
                  <e-column field="Category" label="Category"
type="string"></e-column>
                  <e-column field="SerialNo" label="SerialNo"
type="string"></e-column>
                  <e-column field="InvoiceNo" label="InvoiceNo"
type="string"></e-column>
                  <e-column field="Status" label="Status" type="string"></e-
column>
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: hardwareData,
            importRules: {
        'condition': 'or',
        'rules': [{
            'label': 'Category',
            'field': 'Category',
            'type': 'string',
            'operator': 'equal',
            'value': 'Laptop'
        }]
    }
        };
    }
}
var hardwareData = [{
      'TaskID': 1,
      'Name': 'Lenovo Yoga',
      'Category': 'Laptop',
      'SerialNo': 'CB27932009',
      'InvoiceNo': 'INV-2878',
      'Status': 'Assigned'
  },
  {
      'TaskID': 2,
      'Name': 'Acer Aspire',
      'Category': 'Others',
      'SerialNo': 'CB35728290',
      'InvoiceNo': 'INV-3456',
      'Status': 'In-repair'
  },
  {
      'TaskID': 3,
      'Name': 'Apple MacBook',
      'Category': 'Laptop',
      'SerialNo': 'CB35628728',
      'InvoiceNo': 'INV-2763',
      'Status': 'In-repair'
  }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs14" %}

You can use this property in the mobile mode to restrict the nested group creation.

State persistence in Vue Query builder component

State persistence allows you to maintain the current state in the browser's localStorage even if the browser is refreshed or if you move to the next page within the browser. State persistence stores the Query Builder's rule object in the local storage when the enablePersistence is defined to true.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="dataSource"
:rule="importRules" enablePersistence="true">
                <e-columns>
                    <e-column field="TaskID" label="Task ID"
type="number"></e-column>
                    <e-column field="Name" label="Name" type="string"></e-
column>
                    <e-column field="Category" label="Category"
type="string"></e-column>
                    <e-column field="SerialNo" label="SerialNo"
type="string"></e-column>
                    <e-column field="InvoiceNo" label="InvoiceNo"
type="string"></e-column>
                    <e-column field="Status" label="Status" type="string"></e-
column>
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
    Vue.use(QueryBuilderPlugin);
export default {
    data: function() {
        return {
            dataSource: hardwareData,
            importRules: {
        'condition': 'or',
        'rules': [{
            'label': 'Category',
            'field': 'Category',
            'type': 'string',
            'operator': 'equal',
            'value': 'Laptop'
        },
```

```
        {
        'condition': 'and',
            'rules': [{
                'label': 'Status',
                'field': 'Status',
                'type': 'string',
                'operator': 'notequal',
                'value': 'Pending'
            },
            {
                'label': 'Task ID',
                'field': 'TaskID',
                'type': 'number',
                'operator': 'equal',
                'value': 5675
            }]
        }]
    }
        }
    }
}
var hardwareData = [{
      'TaskID': 1,
      'Name': 'Lenovo Yoga',
      'Category': 'Laptop',
      'SerialNo': 'CB27932009',
      'InvoiceNo': 'INV-2878',
      'Status': 'Assigned'
  },
  {
      'TaskID': 2,
      'Name': 'Acer Aspire',
      'Category': 'Others',
      'SerialNo': 'CB35728290',
      'InvoiceNo': 'INV-3456',
      'Status': 'In-repair'
  },
  {
      'TaskID': 3,
      'Name': 'Apple MacBook',
      'Category': 'Laptop',
      'SerialNo': 'CB35628728',
      'InvoiceNo': 'INV-2763',
      'Status': 'In-repair'
  }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    .e-query-builder {
```

```
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs17" %}

Rtl in Vue Query builder component

RTL provides an option to switch the text direction and layout of the Query Builder component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL, set the [enableRtl] (https://ej2.syncfusion.com/vue/documentation/right-to-left/) to true.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="70%" :dataSource="dataSource"
:rule="importRules" locale="ar-AE" enableRtl="true">
                <e-columns>
                  <e-column field="TaskID" label="Task ID"
type="number"></e-column>
                  <e-column field="Name" label="Name" type="string"></e-
column>
                  <e-column field="Category" label="Category"
type="string"></e-column>
                  <e-column field="SerialNo" label="SerialNo"
type="string"></e-column>
                  <e-column field="InvoiceNo" label="InvoiceNo"
type="string"></e-column>
                  <e-column field="Status" label="Status" type="string"></e-
column>
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { L10n, setCulture } from '@syncfusion/ej2-base';
Vue.use(QueryBuilderPlugin);
setCulture('ar-AE');
L10n.load({
   'ar-AE': {
        'querybuilder': {
            'AddGroup': 'إضافة مجموعة',
            'AddCondition': 'اضافة الشرط',
            'DeleteRule': 'أزل هذا الشرط',
            'DeleteGroup': 'حذف المجموعة',
            'Edit': 'تصحيح',
            'SelectField': 'اختر مجال',
            'DeleteRule': 'أزل هذا الشرط',
            'DeleteGroup': 'حذف المجموعة',
            'SelectOperator': 'حدد المشغل',
            'StartsWith': 'ابدا ب',
```

```
            'EndsWith': 'ينتهي مع',
            'Contains': 'يحتوي على',
            'Equal': 'مساو',
            'NotEqual': 'ليس متساوي',
            'LessThan': 'أقل من',
            'LessThanOrEqual': 'اصغر من او يساوي',
            'GreaterThan': 'أكثر من',
            'GreaterThanOrEqual': 'أكبر من أو يساوي',
            'Between': 'ما بين',
            'NotBetween': 'ليس بينهما',
            'In': 'في',
            'NotIn': 'ليس في',
            'Remove': 'إزالة',
            'ValidationMessage': 'هذه الخانة مطلوبه'
        }
    }
});
export default {
    data: function() {
        return {
            dataSource: hardwareData,
            importRules:{
        'condition': 'or',
        'rules': [{
            'label': 'Category',
            'field': 'Category',
            'type': 'string',
            'operator': 'equal',
            'value': 'Laptop'
        }]
    }
        };
    }
}
var hardwareData = [{
      'TaskID': 1,
      'Name': 'Lenovo Yoga',
      'Category': 'Laptop',
      'SerialNo': 'CB27932009',
      'InvoiceNo': 'INV-2878',
      'Status': 'Assigned'
  },
  {
      'TaskID': 2,
      'Name': 'Acer Aspire',
      'Category': 'Others',
      'SerialNo': 'CB35728290',
      'InvoiceNo': 'INV-3456',
      'Status': 'In-repair'
  },
  {
      'TaskID': 3,
      'Name': 'Apple MacBook',
      'Category': 'Laptop',
      'SerialNo': 'CB35628728',
      'InvoiceNo': 'INV-2763',
      'Status': 'In-repair'
```

```
  }];
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs15" %}

Summary view in Vue Query builder component

Summary view allows you to show or hide the filtered query. By default, the value is false. You can enable by setting the summaryView property to true.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="col-lg-12 querybuilder-control">
            <ejs-querybuilder width="30%" :dataSource="dataSource"
:rule="importRules" :summaryView="true">
                <e-columns>
                    <e-column field='EmployeeID' label='Employee ID'
type='number' />
                    <e-column field='FirstName' label='First Name'
type='string' />
                    <e-column field='TitleOfCourtesy' label='Title Of
Courtesy' type='boolean' :values="values"/>
                    <e-column field='Title' label='Title' type='string' />
                    <e-column field='HireDate' label='Hire Date' type='date'
format='dd/MM/yyyy' />
                    <e-column field='Country' label='Country' type='string'
/>
                    <e-column field='City' label='City' type='string' />
                </e-columns>
            </ejs-querybuilder>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { QueryBuilderPlugin } from "@syncfusion/ej2-vue-querybuilder";
import { employeeData } from './datasource.js';
Vue.use(QueryBuilderPlugin);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    data: function() {
        return {
            dataSource: employeeData,
            values: ['Mr.', 'Mrs.'],
            importRules: {
                'condition': 'and',
    'rules': [{
        'label': 'EmployeeID',
        'field': 'EmployeeID',
        'type': 'number',
        'operator': 'notequal',
        'value': '5'
    },
    {
        'condition': 'or',
        'rules': [{
            'label': 'Title Of Courtesy',
                'field': 'TitleOfCourtesy',
                'type': 'string',
                'operator': 'equal',
                'value': 'Mr.'
        },
        {
            'label': 'Country',
            'field': 'Country',
            'type': 'string',
            'operator': 'equal',
            'value': 'USA'
        },
        {
        'condition': 'and',
            'rules': [{
                'label': 'City',
                'field': 'City',
                'type': 'string',
                'operator': 'equal',
                'value': 'London'
            }]
        }]
    }]
}
    };
}
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-vue-
querybuilder/styles/material.css";
    .e-query-builder {
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/query-builder/default-cs18" %}

RadioButton

Getting Started with the Vue Radio button Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Radio button component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies are required to use the RadioButton component in your application.

```
|-- @syncfusion/ej2-vue-buttons
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-buttons
|-- @syncfusion/ej2-vue-base
```

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

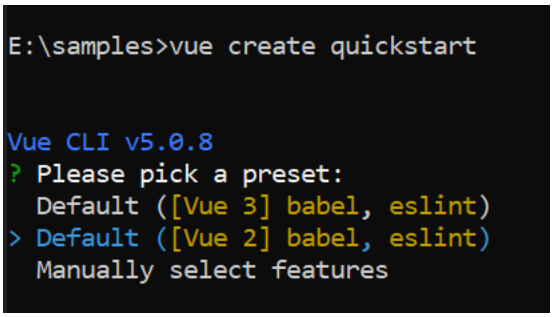
```bash
npm install -g @vue/cli
vue create quickstart
cd quickstart
npm run serve
```

or

```bash
yarn global add @vue/cli
vue create quickstart
cd quickstart
yarn run serve
```

When creating a new project, choose the option `Default ([Vue 2] babel, eslint)` from the menu.

Copyright © 2001 -2024 Syncfusion Inc.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Radio button component as an example. Install the @syncfusion/ej2-vue-buttons package by running the following command:

`bash

npm install @syncfusion/ej2-vue-buttons --save

`

or

`bash

yarn add @syncfusion/ej2-vue-buttons

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Radio button component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Radio button component using Composition API or Options API:

1\. First, import and register the Radio button component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { RadioButtonComponent as EjsRadiobutton } from "@syncfusion/ej2-vue-buttons";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { RadioButtonComponent } from "@syncfusion/ej2-vue-buttons";
export default {
components: {
'ejs-radiobutton': RadioButtonComponent
}
}
</script>
```

2\. In the template section, define the Radio button component with the label property.

~/SRC/APP.VUE

```
<template>
<ejs-radiobutton label='Default'></ejs-radiobutton>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ul>
<li><ejs-radiobutton label='Option 1' name='default'></ejs-radiobutton></li>
<li><ejs-radiobutton label='Option 2' name='default' checked=true></ejs-radiobutton></li>
</ul>
</template>
<script setup>
import { RadioButtonComponent as EjsRadiobutton } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.e-radio-wrapper {
  margin-top: 18px;
}
li {
  list-style: none;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ul>
<li><ejs-radiobutton label='Option 1' name='default'></ejs-radiobutton></li>
<li><ejs-radiobutton label='Option 2' name='default' checked=true></ejs-
radiobutton></li>
</ul>
</template>
<script>
import { RadioButtonComponent } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
    components: {
        'ejs-radiobutton': RadioButtonComponent
    }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.e-radio-wrapper {
  margin-top: 18px;
}
li {
  list-style: none;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/radio-button/default-cs3" %}

Change the RadioButton state

The Essential JS 2 RadioButton contains 2 states visually, they are as follows:

- Checked
- Unchecked

Copyright © 2001 -2024 Syncfusion Inc.

The RadioButton checked property is used to handle the checked and unchecked state. In the checked state an inner circle will be added to the visualization of RadioButton.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ul>
<li><ejs-radiobutton label='Option 1' name='default' checked=true></ejs-
radiobutton></li>
<li><ejs-radiobutton label='Option 2' name='default'></ejs-radiobutton></li>
</ul>
</template>
<script setup>
import { RadioButtonComponent as EjsRadiobutton } from "@syncfusion/ej2-vue-
buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.e-radio-wrapper {
  margin-top: 18px;
}
li {
  list-style: none;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ul>
<li><ejs-radiobutton label='Option 1' name='default' checked=true></ejs-
radiobutton></li>
<li><ejs-radiobutton label='Option 2' name='default'></ejs-radiobutton></li>
</ul>
</template>
<script>
import { RadioButtonComponent } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
  components: {
    'ejs-radiobutton': RadioButtonComponent
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.e-radio-wrapper {
  margin-top: 18px;
}
li {
  list-style: none;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</style>
```

{% previewsample "page.domainurl/code-snippet/radio-button/default-cs4" %}

Getting Started with the Vue RadioButton Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue RadioButton component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

Copyright © 2001 -2024 Syncfusion Inc.

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue RadioButton component as an example. To use the Vue RadioButton component in the project, the @syncfusion/ej2-vue-buttons package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-buttons --save

Copyright © 2001 -2024 Syncfusion Inc.

`

or

`bash

yarn add @syncfusion/ej2-vue-buttons

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the RadioButton component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue RadioButton component using Composition API or Options API:

1.First, import and register the RadioButton component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { RadioButtonComponent as EjsRadiobutton } from "@syncfusion/ej2-vue-buttons";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { RadioButtonComponent } from "@syncfusion/ej2-vue-buttons";
//Component registration
export default {
name: "App",
components: {
"ejs-radiobutton": RadioButtonComponent
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

2.In the template section, define the RadioButton component with the label property and name property and checked definitions.

~/SRC/APP.VUE

```
<template>
<ul>
<li><ejs-radiobutton label='Label1' name='name'></ejs-radiobutton></li>
<li><ejs-radiobutton label='Label2' name='name' checked="isChecked"></ejs-
radiobutton></li>
</ul>
</template>
```

3.Declare the values for the label, name, checked property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const Label1 = "Option 1";
const Label2 = "Option 2";
const name = "default";
bool isChecked = true;
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
Label1 = "Option 1";
Label2 = "Option 2";
name = "default";
isChecked = true;
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ul>
<li><ejs-radiobutton label='Label1' name='name'></ejs-radiobutton></li>
<li><ejs-radiobutton label='Label2' name='name' checked=isChecked></ejs-
radiobutton></li>
</ul>
</template>
<script setup>
import { RadioButtonComponent as EjsRadiobutton } from "@syncfusion/ej2-vue-
buttons";
const Label1 = "Option 1";
const Label2 = "Option 2";
const name = "default";
bool isChecked = true;
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ul>
<li><ejs-radiobutton label='Label1' name='name'></ejs-radiobutton></li>
<li><ejs-radiobutton label='Label2' name='name' checked=isChecked></ejs-
radiobutton></li>
</ul>
</template>
<script>
import { RadioButtonComponent } from "@syncfusion/ej2-vue-buttons";
// Component registration
export default {
name: "App",
// Declaring component and its directives
components: {
"ejs-radiobutton": RadioButtonComponent
},
// Bound properties declarations
data() {
return {
Label1 = "Option 1";
Label2 = "Option 2";
name = "default";
isChecked = true;
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

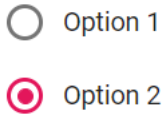
or

`bash

yarn run dev

`

The output will appear as follows:



Sample: vue-3-radio-button-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Label and size in Vue Radio button component

This section explains the different sizes and labels.

Label

RadioButton caption can be defined by using the label property. This reduces the manual addition of label for RadioButton. You can customize the label position before or after the RadioButton through the labelPosition property.

APP.VUE

```
<template>
<ul>
<li> <ejs-radiobutton label="Left Side Label" name="position"
labelPosition="Before"></ejs-radiobutton></li>
<li> <ejs-radiobutton label="Right Side Label" name="position"
checked="true"></ejs-radiobutton></li>
</ul>
</template>
<script>
import Vue from 'vue';
import { RadioButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RadioButtonPlugin);
export default {}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
.e-radio-wrapper {
  margin-top: 18px;
}
li {
  list-style: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/radio-button/default-cs9" %}

Copyright © 2001 -2024 Syncfusion Inc.

Size

The different RadioButton sizes available are default and small. To reduce the size of default RadioButton to small, set the cssClass property to e-small.

APP.VUE

```
<template>
<ul>
<li> <ejs-radiobutton label="Small" name="size" checked="true" cssClass="e-small"></ejs-radiobutton></li>
<li> <ejs-radiobutton label="Default" name="size"></ejs-radiobutton></li>
</ul>
</template>
<script>
import Vue from 'vue';
import { RadioButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RadioButtonPlugin);
export default {}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
.e-radio-wrapper {
  margin-top: 18px;
}
li {
  list-style: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/radio-button/default-cs10" %}

See Also

- How to customize the RadioButton appearance

Accessibility in Vue Radio button component

The Radio button component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Radio button component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Radio button component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Radio button component:

| Attributes | Purpose |

| --- | --- |

| aria-disabled | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Radio button component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Radio button component.

| **Press** | **To do this** |

| --- | --- |

| UP/Left arrow | Move and select the previous options. |

| Down/Right arrow | Move and select the next options. |

Ensuring accessibility

The Radio button component component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Radio button component is shown in the following sample. Open the [sample](https://ej2.syncfusion.com/accessibility/Radio button.html) in a new window to evaluate the accessibility of the Radio button component with accessibility tools.

{% previewsample "page.domainurl/code-snippet/radio-button/default-cs2" %}

See also

- Accessibility in Syncfusion Vue components

Style and appearance in Vue Radio button component

To modify the RadioButton appearance, you need to override the default CSS of RadioButton component. Please find the list of CSS classes and its corresponding section in RadioButton. Also, you have an option to create your own custom theme for the controls using our Theme Studio.

CSS Class | Purpose of Class

|.e-radio-wrapper|To customize the radio button wrapper

|.e-radio + label:hover::before|To customize the radiobutton on hover

|.e-radio:checked + label::after, e-radio:checked + label::before |To customize the checked radiobutton

|.e-radio:checked:focus + label::before, .e-radio:checked + label:hover::before |To customize the checked radiobutton on hover

How To

Customize radiobutton appearance in Vue Radio button component

You can customize the appearance of the RadioButton component by using the CSS rules. Define own CSS rules according to your requirement and assign the class name to the cssClass property.

The background and border color of the RadioButton is customized through the custom classes to create the primary, success, info, warning, and danger type of RadioButton.

APP.VUE

```
<template>
<ul>
<li> <ejs-radiobutton label="Primary" name="custom" cssClass="e-
primary"></ejs-radiobutton></li>
<li> <ejs-radiobutton label="Success" name="custom" cssClass="e-
success"></ejs-radiobutton></li>
<li> <ejs-radiobutton label="Info" name="custom" cssClass="e-info"
checked="true"></ejs-radiobutton></li>
<li> <ejs-radiobutton label="Warning" name="custom" cssClass="e-
warning"></ejs-radiobutton></li>
<li> <ejs-radiobutton label="Danger" name="custom" cssClass="e-
danger"></ejs-radiobutton></li>
</ul>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { RadioButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RadioButtonPlugin);
export default {}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
.e-radio-wrapper {
  margin-top: 18px;
}
li {
  list-style: none;
}
.e-success .e-radio:checked + label::after { /* csslint allow: adjoining-
classes */
  background-color: #689f38;
  border-color: #689f38;
}
.e-success .e-radio:checked:focus + label::after, .e-success .e-
radio:checked + label:hover::after { /* csslint allow: adjoining-classes */
  background-color: #449d44;
  border-color: #449d44;
}
.e-success .e-radio:checked + label::before {
  border-color: #689f38;
}
.e-success .e-radio:checked:focus + label::before, .e-success .e-
radio:checked + label:hover::before { /* csslint allow: adjoining-classes */
  border-color: #449d44;
}
.e-success .e-radio +label:hover::before {
  border-color: #b1afaf
}
.e-info .e-radio:checked + label::after { /* csslint allow: adjoining-
classes */
  background-color: #2196f3;
  border-color: #2196f3;
}
.e-info .e-radio:checked:focus + label::after, .e-info .e-radio:checked +
label:hover::after { /* csslint allow: adjoining-classes */
  background-color: #0b7dda;
  border-color: #0b7dda;
}
.e-info .e-radio:checked + label::before {
  border-color: #2196f3;
}
.e-info .e-radio:checked:focus + label::before, .e-info .e-radio:checked +
label:hover::before {
  border-color: #0b7dda;
}
.e-info .e-radio + label:hover::before {
  border-color: #b1afaf
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.e-warning .e-radio:checked + label::after { /* csslint allow: adjoining-
classes */
  background-color: #ef6c00;
  border-color: #ef6c00;
}
.e-warning .e-radio:checked:focus + label::after, .e-warning .e-
radio:checked + label:hover::after { /* csslint allow: adjoining-classes */
  background-color: #cc5c00;
}
.e-radio:checked + .e-warning::before {
  border-color: #ef6c00;
}
.e-warning .e-radio:checked:focus + label::before, .e-warning .e-
radio:checked + label:hover::before {
  border-color: #cc5c00;
}
.e-warning .e-radio + label:hover::before {
  border-color: #b1afaf
}
.e-danger .e-radio:checked + label::after { /* csslint allow: adjoining-
classes */
  background-color: #d84315;
  border-color: #d84315;
}
.e-danger .e-radio:checked:focus + label::after, .e-danger .e-radio:checked
+ label:hover::after { /* csslint allow: adjoining-classes */
  background-color: #ba330a;
  border-color: #ba330a;
}
.e-danger .e-radio:checked + label::before {
  border-color: #d84315;
}
.e-danger .e-radio:checked:focus + label::before, .e-danger .e-radio:checked
+ label:hover::before {
  border-color: #ba330a;
}
.e-danger .e-radio + label:hover::before {
  border-color: #b1afaf
}
</style>
```

{% previewsample "page.domainurl/code-snippet/radio-button/default-cs5" %}

Name and value in form submit in Vue Radio button component

The name attribute of the RadioButton is used to group RadioButton. When the RadioButton are grouped in form, the checked items value attribute will be post to server on form submit that can be retrieved through the name. The disabled RadioButton value will not be sent to the server on form submit.

In the following code snippet, Credit and Debit card is in checked state. Now, the value that is in checked state will be sent on form submit.

APP.VUE

```
<template>
<form>
```

```
<ul>
<li> <ejs-radiobutton label="Credit/Debit Card" name="payment"
value="credit/debit" checked="true"></ejs-radiobutton></li>
<li> <ejs-radiobutton label="Net Banking" name="payment"
value="netbanking"></ejs-radiobutton></li>
<li> <ejs-radiobutton label="Cash on Delivery" name="payment"
value="cashondelivery"></ejs-radiobutton></li>
<li> <ejs-radiobutton label="Others" name="payment" value="payment"></ejs-
radiobutton></li>
<li> <ejs-button isPrimary="true" >Submit</ejs-button></li>
</ul>
</form>
</template>
<script>
import Vue from 'vue';
import { RadioButtonPlugin, ButtonPlugin } from '@syncfusion/ej2-vue-
buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RadioButtonPlugin);
Vue.use(ButtonPlugin);
export default {}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
.e-radio-wrapper {
  margin-top: 18px;
}
button {
  margin: 20px 0 0 5px;
}
li {
  list-style: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/radio-button/default-cs6" %}

Right to left in Vue Radio button component

RadioButton component has RTL support. This can be achieved by setting enableRtl as true.

The following example illustrates how to enable right-to-left support in RadioButton component.

APP.VUE

```
<template>
<ul>
<li><ejs-radiobutton label='Option 1' name='default' enableRtl="true"></ejs-
radiobutton></li>
<li><ejs-radiobutton label='Option 2' name='default' enableRtl="true"
checked=true></ejs-radiobutton></li>
</ul>
</template>
<script>
import Vue from 'vue';
import { RadioButtonPlugin } from "@syncfusion/ej2-vue-buttons";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RadioButtonPlugin);
export default {}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.e-radio-wrapper {
  margin-top: 18px;
}
li {
  list-style: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/radio-button/default-cs7" %}

Set the disabled state in Vue Radio button component

RadioButton component can be enabled/disabled by giving disabled property. To disable RadioButton component, the disabled property can be set as true.

The following example illustrates how to disable a radio button and the selected one is displayed using change event.

APP.VUE

```
<template>
<ul>
<li><div id="text">Selected: Option 1</div></li>
<li> <ejs-radiobutton label="Option 1" name="default" checked="true"
:change="onChange" ></ejs-radiobutton></li>
<li> <ejs-radiobutton label="Option 2" name="default" disabled="true"
:change="onChange"></ejs-radiobutton></li>
<li> <ejs-radiobutton label="Option 3" name="default"
:change="onChange"></ejs-radiobutton></li>
</ul>
</template>
<script>
import Vue from 'vue';
import { RadioButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RadioButtonPlugin);
export default {
  methods: {
    onChange: function(args) {
      document.getElementById('text').innerText = 'Selected : ' +
args.event.target.ej2_instances[0].label;
    }
  }
};
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.e-radio-wrapper {
  margin-top: 18px;
}
li {
  list-style: none;
}
#text {
  font-size: 16px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/radio-button/default-cs8" %}

Range Navigator

Getting Started with the Vue Range navigator Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Range navigator component

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

Below is the list of minimum dependencies required to use the range navigator component.

`javascript

|-- @syncfusion/ej2-vue-charts

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-data

|-- @syncfusion/ej2-pdf-export

|-- @syncfusion/ej2-file-utils

|-- @syncfusion/ej2-compression

|-- @syncfusion/ej2-navigations

|-- @syncfusion/ej2-calendars

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-svg-base

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

Copyright © 2001 -2024 Syncfusion Inc.

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

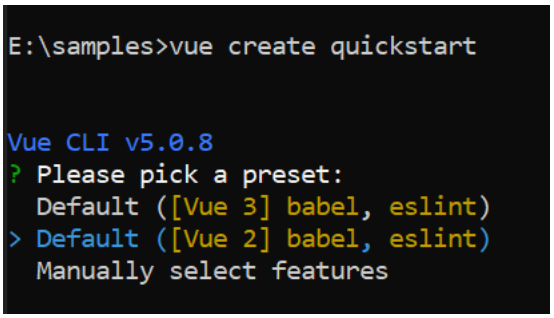
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Range navigator component as an example. Install the @syncfusion/ej2-vue-charts package by running the following command:

`bash

npm install @syncfusion/ej2-vue-charts --save

`

or

`bash

yarn add @syncfusion/ej2-vue-charts

`

The **--save** will instruct NPM to include the range navigator package inside of the dependencies section of the package.json.

Copyright © 2001 -2024 Syncfusion Inc.

Add Syncfusion Vue component

Follow the below steps to add the Vue Message component:

1\. First, import and register the Message component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { RangeNavigatorComponent } from '@syncfusion/ej2-vue-charts';
export default {
components: {
'ejs-rangenavigator': RangeNavigatorComponent
}
}
</script>
```

2\. In the template section, define Range navigator component.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-rangenavigator></ejs-rangenavigator>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

~/SRC/APP.VUE

```
<template>
  <div id="app">
      <ejs-rangenavigator></ejs-rangenavigator>
  </div>
</template>
<script>
import { RangeNavigatorComponent } from '@syncfusion/ej2-vue-charts';
export default {
  components: {
    'ejs-rangenavigator': RangeNavigatorComponent
  },
  data () {
    return {
    }
  }
}
</script>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

Copyright © 2001 -2024 Syncfusion Inc.

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs9" %}

Module Injection

To create range navigator with additional features, inject the required modules. The following modules are used to extend rangenavigator's basic functionality.

•AreaSeries - Inject this module to use area series.

•DateTime - Inject this module to use date time axis.

•RangeTooltip - Inject this module to show the tooltip.

These modules should be injected to the provide section as follows,

`javascript

import { RangeNavigatorComponent, AreaSeries, DateTime, RangeTooltip } from "@syncfusion/ej2-vue-charts";

export default {

components: {

'ejs-rangenavigator': RangeNavigatorComponent

},

provide: {

rangeNavigator: [AreaSeries, DateTime, RangeTooltip]

}

};

</script>

`

Populate Range Navigator with Data

Now, we are going to provide data to the range navigator. Add a series object to the range navigator by using series property. Now map the field names x and y in the JSON data to the xName and yName properties of the series, then set the JSON data to dataSource property. Since the JSON contains Datetime data, set the valueType range Navigator to Category. By default, the axis valueType is Numeric.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat'>
            <e-rangenavigator-series-collection>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import { RangeNavigatorComponent, RangenavigatorSeriesDirective,
RangenavigatorSeriesCollectionDirective, AreaSeries, DateTime } from
"@syncfusion/ej2-vue-charts";
import { bitCoinData } from "./default_data.js";
export default {
  components: {
    'ejs-rangenavigator': RangeNavigatorComponent,
    'e-rangenavigator-series': RangenavigatorSeriesDirective,
    'e-rangenavigator-series-collection':
RangenavigatorSeriesCollectionDirective
  },
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs10" %}

Enable Tooltip

The tooltip is useful to show the selected data. You can enable tooltip by setting the `enable` property as true in tooltip object and by injecting `RangeTooltip` module using `provide` method.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' :tooltip='tooltip'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { RangeNavigatorComponent,
RangenavigatorSeriesDirective,RangenavigatorSeriesCollectionDirective,
AreaSeries, DateTime, RangeTooltip } from "@syncfusion/ej2-vue-charts";
import { bitCoinData } from "./default_data.js";
export default {
  components: {
    'ejs-rangenavigator': RangeNavigatorComponent,
    'e-rangenavigator-series': RangenavigatorSeriesDirective,
    'e-rangenavigator-series-collection':
RangenavigatorSeriesCollectionDirective
  },
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     tooltip: { enable: true, displayMode: "Always", format: "MM/dd/yyyy" },
     labelFormat: 'MMM-yy',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries, RangeTooltip]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs11" %}

Getting Started with the Vue Range Navigator Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Range Navigator component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

Copyright © 2001 -2024 Syncfusion Inc.

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Range Navigator component as an example. To use the Vue Range Navigator component in the project, the @syncfusion/ej2-vue-charts package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-charts --save

`

or

`bash

yarn add @syncfusion/ej2-vue-charts

`

Add Syncfusion Vue component

Follow the below steps to add the Vue Range Navigator component using Composition API or Options API:

1.First, import and register the Range Navigator component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { RangeNavigatorComponent as EjsRangenavigator, StepLineSeries,
DateTime, RangenavigatorSeriesCollectionDirective as
ERangenavigatorSeriesCollection, RangenavigatorSeriesDirective as
ERangenavigatorSeries} from "@syncfusion/ej2-vue-charts";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { RangeNavigatorComponent, StepLineSeries, DateTime,
RangenavigatorSeriesCollectionDirective,
RangenavigatorSeriesDirective } from "@syncfusion/ej2-vue-charts";
//Component registration
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
name: "App",
components: {
"ejs-rangenavigator": RangeNavigatorComponent,
"e-rangenavigator-series-collection":
RangenavigatorSeriesCollectionDirective,
"e-rangenavigator-series":  RangenavigatorSeriesDirective
}
}
</script>
```

2.In the template section, define the Range Navigator component with the dataSource property.

~/SRC/APP.VUE

```
<template>
<ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat'>
<e-rangenavigator-series-collection>
<e-rangenavigator-series :dataSource='data' type='StepLine' xName='Date'
yName='Close' width=2>
</e-rangenavigator-series>
</e-rangenavigator-series-collection>
</ejs-rangenavigator>
</template>
```

3.Declare the values for the dataSource property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const data = [
{ Date: new Date("2005-01-01"), Close: 21 },
{ Date: new Date("2006-01-01"), Close: 24 },
{ Date: new Date("2007-01-01"), Close: 36 },
{ Date: new Date("2008-01-01"), Close: 38 },
{ Date: new Date("2009-01-01"), Close: 54 },
{ Date: new Date("2010-01-01"), Close: 57 },
{ Date: new Date("2011-01-01"), Close: 62 }
];
const valueType = 'DateTime';
const value = [new Date("2008-01-01"), new Date("2010-01-01")];
const labelFormat = 'MMM-yy';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
data: [
{ Date: new Date("2005-01-01"), Close: 21 },
{ Date: new Date("2006-01-01"), Close: 24 },
{ Date: new Date("2007-01-01"), Close: 36 },
{ Date: new Date("2008-01-01"), Close: 38 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ Date: new Date("2009-01-01"), Close: 54 },
{ Date: new Date("2010-01-01"), Close: 57 },
{ Date: new Date("2011-01-01"), Close: 62 }
],
valueType: 'DateTime',
value: [new Date("2008-01-01"), new Date("2010-01-01")],
labelFormat: 'MMM-yy'
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat'>
<e-rangenavigator-series-collection>
<e-rangenavigator-series :dataSource='data' type='StepLine' xName='Date'
yName='Close' width=2>
</e-rangenavigator-series>
</e-rangenavigator-series-collection>
</ejs-rangenavigator>
</template>
<script setup>
import { provide } from 'vue';
import { RangeNavigatorComponent as EjsRangenavigator, StepLineSeries,
DateTime, RangenavigatorSeriesCollectionDirective as
ERangenavigatorSeriesCollection, RangenavigatorSeriesDirective as
ERangenavigatorSeries} from "@syncfusion/ej2-vue-charts";
const data = [
{ Date: new Date("2005-01-01"), Close: 21 },
{ Date: new Date("2006-01-01"), Close: 24 },
{ Date: new Date("2007-01-01"), Close: 36 },
{ Date: new Date("2008-01-01"), Close: 38 },
{ Date: new Date("2009-01-01"), Close: 54 },
{ Date: new Date("2010-01-01"), Close: 57 },
{ Date: new Date("2011-01-01"), Close: 62 }
];
const valueType = 'DateTime';
const value = [new Date("2008-01-01"), new Date("2010-01-01")];
const labelFormat = 'MMM-yy';
const rangeNavigator = [ DateTime, StepLineSeries ];
provide('rangeNavigator', rangeNavigator);
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat'>
<e-rangenavigator-series-collection>
<e-rangenavigator-series :dataSource='data' type='StepLine' xName='Date'
yName='Close' width=2>
</e-rangenavigator-series>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-rangenavigator-series-collection>
</ejs-rangenavigator>
</template>
<script>
import { RangeNavigatorComponent, StepLineSeries, DateTime,
RangenavigatorSeriesCollectionDirective,
RangenavigatorSeriesDirective } from "@syncfusion/ej2-vue-charts";
//Component registration
export default {
name: "App",
components: {
"ejs-rangenavigator": RangeNavigatorComponent,
"e-rangenavigator-series-collection":
RangenavigatorSeriesCollectionDirective,
"e-rangenavigator-series":  RangenavigatorSeriesDirective
},
data() {
return {
data: [
{ Date: new Date("2005-01-01"), Close: 21 },
{ Date: new Date("2006-01-01"), Close: 24 },
{ Date: new Date("2007-01-01"), Close: 36 },
{ Date: new Date("2008-01-01"), Close: 38 },
{ Date: new Date("2009-01-01"), Close: 54 },
{ Date: new Date("2010-01-01"), Close: 57 },
{ Date: new Date("2011-01-01"), Close: 62 }
],
valueType: 'DateTime',
value: [new Date("2008-01-01"), new Date("2010-01-01")],
labelFormat: 'MMM-yy'
};
},
provide: {
rangeNavigator: [ DateTime, StepLineSeries ]
},
};
</script>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

Copyright © 2001 -2024 Syncfusion Inc.

Sample: vue-3-range-navigator-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Selecting range in Vue Range navigator component

The Range Selector's left and right thumbs are used to indicate the selected range in the large collection of data. A range can be selected in the following ways:

- By dragging the thumbs.
- By tapping on the labels.
- By setting the start and the end through the value property.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
   }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs25" %}

Lightweight in Vue Range navigator component

By default, when the dataSource for series is empty, a lightweight Range Selector will be shown without Chart.

APP.VUE

```
<template>
    <div id="app">
      <ejs-rangenavigator :valueType='valueType'
:intervalType='intervalType'
        :value='value' :labelFormat='labelFormat' :dataSource='dataSource'
        xName='x' yName='y'>
      </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-vue-charts";
import { GetDateTimeData  } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: "DateTime",
      intervalType: "Months",
      labelFormat: "MMM",
      value: [new Date('2018-06-01'), new Date('2018-07-01')],
      dataSource: GetDateTimeData(new Date('2018-01-01'), new Date('2019-01-01')),
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs19" %}

See Also

- Period Selector

Series types in Vue Range navigator component

To render the data, the Range Selector supports three types of series.

<!-- markdownlint-disable MD036 -->

Copyright © 2001 -2024 Syncfusion Inc.

Line

<!-- markdownlint-disable MD036 -->

To render a line series, set the seriestype to Line, and inject the LineSeries module using the RangeNavigator.Inject(LineSeries) method. To render a line series, use series type as **Line** and inject the LineSeries module using RangeNavigator.Inject(LineSeries) method. By default, the line series is rendered in the Range Selector.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' :tooltip='tooltip'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Line'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, LineSeries, DateTime, RangeTooltip } from
"@syncfusion/ej2-vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     tooltip: { enable: true },
     labelFormat: 'MMM-yy',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, LineSeries, RangeTooltip]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs26" %}

Area

To render a step line series, use series type as Area and inject AreaSeries module using RangeNavigator.Inject(AreaSeries) method. To render an area series, use series type as **Area** and inject AreaSeries module using RangeNavigator.Inject(AreaSeries) method.

APP.VUE

```
<template>
    <div id="app">
```

```
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' :tooltip='tooltip'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime, RangeTooltip } from
"@syncfusion/ej2-vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     tooltip: { enable: true },
     labelFormat: 'MMM-yy',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries, RangeTooltip]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs27" %}

StepLine

To render a Step line series, use series type as **Step Line** and inject StepLineSeries module using RangeNavigator.Inject(StepLineSeries) method.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' :tooltip='tooltip'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='StepLine'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, StepLineSeries, DateTime, RangeTooltip } from
"@syncfusion/ej2-vue-charts";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     tooltip: { enable: true },
     labelFormat: 'MMM-yy',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, StepLineSeries, RangeTooltip]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs28" %}

Period selector in Vue Range navigator component

The period selector allows to select a range with specified periods.

Periods

An array of objects that allows the users to specify pre-defined time intervals. The interval property specifies the count value of the button, and the text property specifies the text to be displayed on the button. The intervaltype property allows the users to customize the interval type, and it supports the following types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat'
        :periodSelectorSettings='periodSelectorSettingsTop' >
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime, PeriodSelector } from
"@syncfusion/ej2-vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
      periodSelectorSettingsTop: {
        periods: [
          { text: "1M", interval: 1, intervalType: "Months" },
          { text: "3M", interval: 3, intervalType: "Months" },
          { text: "6M", interval: 6, intervalType: "Months" },
          { text: "1Y", interval: 1, intervalType: "Years" },
          { text: "2Y", interval: 2, intervalType: "Years", selected: true
},
          { text: "ALL" }
        ],
        position: "Top"
       },
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries, PeriodSelector]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs20" %}

To use the period selector feature, inject the PeriodSelector module using the RangeNavigator.Inject(PeriodSelector) method.

Positioning period selector

The position property allows the users to position the period selector at the **Top** or **Bottom**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat'
        :periodSelectorSettings='periodSelectorSettingsTop' >
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime, PeriodSelector } from
"@syncfusion/ej2-vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
      periodSelectorSettingsTop: {
        periods: [
          { text: "1M", interval: 1, intervalType: "Months" },
          { text: "3M", interval: 3, intervalType: "Months" },
          { text: "6M", interval: 6, intervalType: "Months" },
          { text: "1Y", interval: 1, intervalType: "Years" },
          { text: "2Y", interval: 2, intervalType: "Years", selected: true
},
          { text: "ALL" }
        ],
        position: "Bottom"
       },
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries, PeriodSelector]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs21" %}

Height

The height property allows the users to specify the height of the period selector. The default value of the height property is **43px**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat'
        :periodSelectorSettings='periodSelectorSettingsTop' >
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
```

```
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime, PeriodSelector } from
"@syncfusion/ej2-vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
      periodSelectorSettingsTop: {
        periods: [
          { text: "1M", interval: 1, intervalType: "Months" },
          { text: "3M", interval: 3, intervalType: "Months" },
          { text: "6M", interval: 6, intervalType: "Months" },
          { text: "1Y", interval: 1, intervalType: "Years" },
          { text: "2Y", interval: 2, intervalType: "Years", selected: true
},
          { text: "ALL" }
        ],
        position: "Top",
        height: 65
      },
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries, PeriodSelector]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs22" %}

Visibility of range navigator

The disableRangeSelector property allows the users to display only the period selector and not the Range Selector.

APP.VUE

```
<template>
    <div id="app">
    <ejs-rangenavigator valueType='DateTime'
        labelPosition='Outside' :dataSource='data' xName='x' yName='Close'
        :periodSelectorSettings='periodSelectorSettingsTop'
disableRangeSelector=true>
       </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, PeriodSelector, DateTime } from
"@syncfusion/ej2-vue-charts";
import { chartData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data() {
    return {
     valueType: 'DateTime',
      periodSelectorSettingsTop: {
        periods: [
          { text: "1M", interval: 1, intervalType: "Months" },
          { text: "3M", interval: 3, intervalType: "Months" },
          { text: "6M", interval: 6, intervalType: "Months" },
          { text: "1Y", interval: 1, intervalType: "Years" },
          { text: "2Y", interval: 2, intervalType: "Years", selected: true
},
          { text: "ALL" }
        ],
        position: "Top"
      },
     data: chartData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries, PeriodSelector]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs23" %}

See Also

- LightWeight

Labels in Vue Range navigator component

Multilevel labels

The multi-level labels for the Range Selector can be enabled by setting the enableGrouping property to **true**. This is restricted to the DateTime axis alone.

APP.VUE

```
<template>
    <div id="app">
      <ejs-rangenavigator :valueType='valueType'
:intervalType='intervalType' enableGrouping=true
        :value='value' :dataSource='dataSource'
        xName='x' yName='y'>
      </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { GetDateTimeData  } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    let points = [];
    let value = 0;
    let point = {};
    for (let j = 1; j < 1090; j++) {
      value += Math.random() * 10 - 5;
      value = value < 0 ? Math.abs(value) : value;
      point = { x: new Date(2000, 0, j), y: value, z: value + 10 };
      points.push(point);
    }
    return {
     valueType: "DateTime",
      intervalType: "Quarter",
      value: [new Date('2001-01-01'), new Date('2002-01-01')],
      dataSource: points,
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs14" %}

Grouping

The multi-level labels can be grouped using the groupBy property with the following interval types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

APP.VUE

```
<template>
    <div id="app">
      <ejs-rangenavigator :valueType='valueType'
:intervalType='intervalType' enableGrouping=true groupBy='Years'
        :value='value' :dataSource='dataSource'
        xName='x' yName='y'>
      </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { GetDateTimeData  } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data() {
    let points = [];
    let value = 0;
    let point = {};
    for (let j = 1; j < 1090; j++) {
      value += Math.random() * 10 - 5;
      value = value < 0 ? Math.abs(value) : value;
      point = { x: new Date(2000, 0, j), y: value, z: value + 10 };
      points.push(point);
    }
    return {
     valueType: "DateTime",
      intervalType: "Quarter",
      value: [new Date('2001-01-01'), new Date('2002-01-01')],
      dataSource: points,
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs15" %}

Smart labels

The labelIntersectAction property is used to avoid overlapping of labels. The following code sample shows the setting of labelIntersectAction property to **Hide**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' labelIntersectAction='Hide'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat:'y/M/d',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs16" %}

Label positioning

By default, the labels can be placed outside the Range Selector. It can also be placed inside the Range Selector using the labelPosition property.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' labelPosition='Inside'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs17" %}

Labels customization

The font size, color, family, etc. can be customized using the labelStyle setting.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div id="app">
      <ejs-rangenavigator :valueType='valueType'
:intervalType='intervalType' :labelStyle='labelStyle'
        :value='value' :labelFormat='labelFormat' :dataSource='dataSource'
        xName='x' yName='y'>
      </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { GetDateTimeData  } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: "DateTime",
      intervalType: "Months",
      labelFormat: "MMM",
      labelStyle: { color: 'red', size:'10px'}
      value: [new Date('2018-06-01'), new Date('2018-07-01')],
      dataSource: GetDateTimeData(new Date('2018-01-01'), new Date('2019-01-
01')),
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs18" %}

Grid tick in Vue Range navigator component

Grid line customization

The gridlines indicate axis divisions by drawing the chart plot. Gridlines include helpful cues to the user, particularly for large or complicated charts. The width, color, and dashArray of the major gridlines can be customized by using the majorGridLines setting.

APP.VUE

```
<template>
    <div id="app">
    <ejs-rangenavigator :value='value' labelPosition ='Outside'
:tooltip='tooltip' :majorGridLines='majorGridLines' >
            <e-rangenavigator-series-collection >
                <e-rangenavigator-series :dataSource='data' xName='xData'
type='Line' yName='yData' ></e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-vue-charts";
import {  sl } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
    majorGridLines:{
        width: 4,
        color: 'blue',
        dashArray: '5,5'
    },
     tooltip: { enable: true },
     value:[25,40],
     data:  [
      { xData: 10, yData: 35 }, { xData: 20, yData: 28 },
      { xData: 30, yData: 34 }, { xData: 40, yData: 32 },
      { xData: 50, yData: 40 }
     ]
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs12" %}

Tick line customization

Ticklines are the small lines which is drawn on the axis line representing the axis labels. Ticklines will be drawn outside the axis by default. The `width`, `color`, and `dashArray` of the major ticklines can be customized by using the majorTickLines setting.

APP.VUE

```
<template>
    <div id="app">
    <ejs-rangenavigator :value='value' labelPosition ='Outside'
:tooltip='tooltip' :majorTickLines='majorTickLines' >
            <e-rangenavigator-series-collection >
                <e-rangenavigator-series :dataSource='data' xName='xData'
type='Line' yName='yData' ></e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-vue-charts";
import {  sl } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    return {
    majorTickLines:{
       width: 3,
       color: 'red'
   },
     tooltip: { enable: true },
     value:[25,40],
     data:  [
      { xData: 10, yData: 35 }, { xData: 20, yData: 28 },
      { xData: 30, yData: 34 }, { xData: 40, yData: 32 },
      { xData: 50, yData: 40 }
     ]
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs13" %}

Customization in Vue Range navigator component

Navigator appearance

The Range Selector can be customized by using the navigatorStyleSettings. The selectedRegionColor property is used to specify the color for the selected region, whereas the unselectedRegionColor property is used to specify the color for the unselected region.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' :navigatorStyleSettings='navigatorStyleSettings'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     navigatorStyleSettings: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        unselectedRegionColor: 'skyblue',
        selectedRegionColor: 'pink'
         },
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs1" %}

Thumb

The thumb property allows to customize the border, fill color, size, and type of thumb. Thumbs can be of two shapes: **Circle** and **Rectangle**.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' :navigatorStyleSettings='navigatorStyleSettings'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     navigatorStyleSettings: {
     thumb:{
           type: 'Rectangle',
           border: { width: 2, color: 'red'},
           fill: 'pink'
       }
         },
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
```

```
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs2" %}

Border customization

Using the navigatorBorder, the width and color of the Range Selector border can be customized.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' :navigatorBorder='navigatorBorder'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     navigatorBorder : { width: 4, color: 'green'},
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs3" %}

Deferred update

If the enableDeferredUpdate property is set to **true**, then the changed event will be triggered after dragging the slider. If the enableDeferredUpdate is **false**, then the changed event will be triggered when dragging the slider. By default, the enableDeferredUpdate is set to **false**.

APP.VUE

```
<template>
    <div id="app">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' :navigatorBorder='navigatorBorder'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     navigatorBorder : { width: 4, color: 'green'},
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs4" %}

Allow snapping

The allowSnapping property toggles the placement of the slider exactly to the left or on the nearest interval.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' :allowSnapping='allowSnapping'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
      allowSnapping: true,
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs5" %}

Animation

The speed of the animation can be controlled using the animationDuration property. The default value of the animationDuration property is **500** milliseconds.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat'  animationDuration=2000>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs6" %}

See Also

- Grid and Tick Lines
- Labels

Tool tip in Vue Range navigator component

<!-- markdownlint-disable MD036 -->

The tooltip for sliders are supported by the Range Selector. Sliders are used in the Range Selector to select data from a specific range. The tooltip displays the selected start and end values.

<!-- markdownlint-disable MD013 -->

Customization

Tooltip can be customized using the following properties:

- enable - Customizes the visibility of the tooltip.
- fill - Customizes the background color of the tooltip.
- opacity - Customizes the opacity of the tooltip.
- textStyle - Customizes the font size, color, family, style, weight, alignment, and overflow of the tooltip.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' :tooltip='tooltip'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime, RangeTooltip } from
"@syncfusion/ej2-vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     tooltip: { enable: true, displayMode: 'Always', fill:'red', opacity:
0.6, textStyle:{ style: 'Italic', color:'blue', size:'12px'} },
```

```
      labelFormat: 'MMM-yy',
      data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries, RangeTooltip]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs29" %}

Label Format

You can format and parse the date to all globalize format using labelFormat property in an axis. The labelFormat property in the tooltip is used to format and parse the date to all globalize formats.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:tooltip='tooltip'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime, RangeTooltip } from
"@syncfusion/ej2-vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
      valueType: 'DateTime',
      value: [new Date('2017-09-01'), new Date('2018-02-01')],
      tooltip: { enable: true, displayMode: 'Always', labelFormat: 'y/M/d' },
      data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries, RangeTooltip]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs30" %}

The following table shows the results of applying some common date and time formats to the labelFormat property.

Copyright © 2001 -2024 Syncfusion Inc.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format Property Value	Result	Description
new Date(2000, 03, 10)	EEEE	Monday	The Date is displayed in day format
new Date(2000, 03, 10)	yMd	04/10/2000	The Date is displayed in month/date/year format
new Date(2000, 03, 10)	MMM	Apr	The Shorthand month for the date is displayed
new Date(2000, 03, 10)	hm	12:00 AM	Time of the date value is displayed as label
new Date(2000, 03, 10)	hms	12:00:00 AM	The Label is displayed in hours:minutes:seconds format

R t l in Vue Range navigator component

The Range Selector supports right-to-left (RTL), which can be enabled with the enableRtl property.

APP.VUE

```
<template>
    <div id="app">
        <ejs-rangenavigator :valueType='valueType' :value='value'
:labelFormat='labelFormat' :enableRtl='enableRtl'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     enableRtl: true,
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs24" %}

Export print in Vue Range navigator component

Export

The rendered Range Selector can be exported to **JPEG**, **PNG**, **SVG**, or **PDF** format by using the export method in the Range Selector. This method contains the following parameters:

- **Type** - To specify the export type. The component can be exported to **JPEG**, **PNG**, **SVG**, or **PDF** format.
- **File name** - To specify the file name to export.
- **Orientation** - To specify the orientation type. This is applicable only for PDF export type.

APP.VUE

```
<template>
    <div id="app">
     <ejs-button id='export' @click.native='exportIcon'>Export</ejs-button>
        <ejs-rangenavigator ref="chart" :valueType='valueType'
:value='value' :labelFormat='labelFormat'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
Vue.use(ButtonPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  },
  methods: {
    exportIcon: function() {
        this.$refs.chart.export('PNG', 'export');
    }
```

```
    }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs7" %}

Print

The rendered Range Selector can be printed directly from the browser by calling the public method print.

APP.VUE

```
<template>
    <div id="app">
      <ejs-button id='print' @click.native='print'>Print</ejs-button>
        <ejs-rangenavigator ref="chart" :valueType='valueType'
:value='value' :labelFormat='labelFormat'>
            <e-rangenavigator-series-collection>
                <e-rangenavigator-series :dataSource='data' type='Area'
xName='x' yName='y' width=2>
                </e-rangenavigator-series>
            </e-rangenavigator-series-collection>
        </ejs-rangenavigator>
    </div>
</template>
<script>
import Vue from "vue";
import { RangeNavigatorPlugin, AreaSeries, DateTime } from "@syncfusion/ej2-
vue-charts";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { bitCoinData } from "./default_data.js";
Vue.use(RangeNavigatorPlugin);
Vue.use(ButtonPlugin);
export default {
  data() {
    return {
     valueType: 'DateTime',
     value: [new Date('2017-09-01'), new Date('2018-02-01')],
     labelFormat: 'MMM-yy',
     data: bitCoinData
    };
  },
  provide: {
    rangeNavigator: [DateTime, AreaSeries]
  },
  methods: {
    print: function() {
        this.$refs.chart.print();
    }
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/rangenavigator/getting-started-cs8" %}

Copyright © 2001 -2024 Syncfusion Inc.

Accessibility in Vue Range navigator component

The Range navigator component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Range navigator component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | |

| Section 508 Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| Accessibility Checker Validation | |

| Axe-core Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

 - All features of the component meet the requirement.

 - Some features of the component do not meet the requirement.

 - The component does not meet the requirement.

Copyright © 2001 -2024 Syncfusion Inc.

WAI-ARIA attributes

The Range navigator component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Range navigator component:

- region (role)
- aria-label (attribute)

Keyboard interaction

The Range navigator component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Range navigator component.

| **Press** | **To do this** |

| --- | --- |

| Tab | Moves the focus to the Range navigator element. |

| Ctrl + P | Prints the Range navigator. |

Ensuring accessibility

The Range navigator component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Range navigator component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Range navigator component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/range-navigator.html" %}

See also

- Accessibility in Syncfusion Vue components

Range Slider

Getting Started with the Vue Range slider Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Range slider component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

Copyright © 2001 -2024 Syncfusion Inc.

npm run serve

`

or

`bash

yarn global add @vue/cli

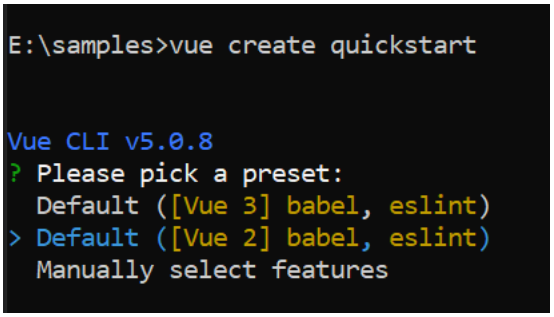
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the [Vue Range slider component](https://www.syncfusion.com/vue-components/vue-slider as an example. Install the @syncfusion/ej2-vue-inputs package by running the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

Copyright © 2001 -2024 Syncfusion Inc.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Range slider component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Range slider component using Composition API or Options API:

1\. First, import and register the Range slider component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SliderComponent as EjsSlider } from "@syncfusion/ej2-vue-inputs";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SliderComponent } from "@syncfusion/ej2-vue-inputs";
export default {
components: {
'ejs-slider': SliderComponent
}
}
</script>
```

2\. In the template section, define the Range slider component with the value property.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-slider id='default' :value='value'></ejs-slider>
</div>
</template>
```

3\. Declare the value for the value property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const value = 30;
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
value: 30
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
    <ejs-slider id='default' :value='value'></ejs-slider>
  </div>
</template>
<script setup>
import { SliderComponent as EjsSlider } from "@syncfusion/ej2-vue-inputs";
const value = 30;
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    color: #008cff;
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
    <ejs-slider id='default' :value='value'></ejs-slider>
  </div>
</template>
<script>
import { SliderComponent } from "@syncfusion/ej2-vue-inputs";
export default {
  components: {
    'ejs-slider': SliderComponent
  },
  data() {
    return {
      value: 30
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    color: #008cff;
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/range-slider/getting-started-cs1" %}

Types

The types of Slider are as follows:

| **Types** | **Usage** |

| --- | --- |

| Default | Shows a default Slider to select a single value. |

| MinRange | Displays the shadow from the start value to the current selected value. |

| Range | Selects a range of values. It also displays the shadow in-between the selection range. |

Both the Default Slider and Min-Range Slider have same behavior that is used to select a single value. In Min-Range Slider, a shadow is considered from the start value to current handle position. But the Range Slider contains two handles that is used to select a range of values and a shadow is considered in between the two handles.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
```

```
    <div id="app">
     <label class="labeltext">Default</label>
     <ejs-slider id='default' :value='minvalue'></ejs-slider>
      <label class="labeltext">MinRange</label>
      <ejs-slider id='default' :value='minvalue' :type="mintype"></ejs-
slider>
     <label class="labeltext">Range</label>
    <ejs-slider id='type' :value='value' :type="type"></ejs-slider>
  </div>
</template>
<script setup>
import { SliderComponent as EjsSlider } from "@syncfusion/ej2-vue-inputs";
const value = [30, 70];
const type = 'Range';
const mintype = 'MinRange';
const minvalue = 30;
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
     <label class="labeltext">Default</label>
     <ejs-slider id='default' :value='minvalue'></ejs-slider>
      <label class="labeltext">MinRange</label>
      <ejs-slider id='default' :value='minvalue' :type="mintype"></ejs-
slider>
     <label class="labeltext">Range</label>
    <ejs-slider id='type' :value='value' :type="type"></ejs-slider>
  </div>
</template>
<script>
import { SliderComponent } from "@syncfusion/ej2-vue-inputs";
export default {
  components: {
    'ejs-slider': SliderComponent
  },
  data() {
    return {
       value: [30, 70],
       type: 'Range',
       mintype: 'MinRange',
       minvalue: 30
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/getting-started-cs2" %}

Customization

Orientation

The Slider can be displayed, either in horizontal or vertical orientation. By default, the Slider renders in horizontal orientation.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
    <ejs-slider id='orientation' :value='value'
:orientation="orientation"></ejs-slider>
  </div>
</template>
<script setup>
import { SliderComponent as EjsSlider } from "@syncfusion/ej2-vue-inputs";
// vertical orientation
const orientation = 'Vertical';
const value = 30;
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    color: #008cff;
    height: 340px;
    left: 30%;
    position: absolute;
    width: 50%;
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div id="app">
    <ejs-slider id='orientation' :value='value'
:orientation="orientation"></ejs-slider>
  </div>
</template>
<script>
import { SliderComponent } from "@syncfusion/ej2-vue-inputs";
export default {
  components: {
    'ejs-slider': SliderComponent
  },
  data() {
    return {
        // vertical orientation
        orientation: 'Vertical',
        value: 30
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    color: #008cff;
    height: 340px;
    left: 30%;
    position: absolute;
    width: 50%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/getting-started-cs3" %}

Tooltip

The Slider displays the tooltip to indicate the current value by clicking the Slider bar or drag the Slider handle. The Tooltip position can be customized by using the placement property. Also decides the tooltip display mode on a page, i.e., on hovering, focusing, or clicking on the Slider handle. Tooltip always remains/displays on the page.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
    <ejs-slider id='tooltip' :value='value' :tooltip="tooltip"
:type="type"></ejs-slider>
  </div>
</template>
<script setup>
import { SliderComponent as EjsSlider } from "@syncfusion/ej2-vue-inputs";
// Slider tooltip
const tooltip = { placement: 'After', isVisible: true, showOn: 'Always' };
const value = 30;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
const type = 'MinRange';
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    color: #008cff;
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
    <ejs-slider id='tooltip' :value='value' :tooltip="tooltip"
:type="type"></ejs-slider>
  </div>
</template>
<script>
import { SliderComponent } from "@syncfusion/ej2-vue-inputs";
export default {
  components: {
    'ejs-slider': SliderComponent
  },
  data() {
    return {
        // Slider tooltip
        tooltip: { placement: 'After', isVisible: true, showOn: 'Always' },
        value: 30,
        type: 'MinRange'
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    color: #008cff;
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/range-slider/getting-started-cs4" %}

Buttons

The Slider value can be changed by using the Increase and Decrease buttons. In Range Slider, by default the first handle value will be changed while clicking the button. Change the handle focus and press the button to change the last focused handle value.

After enabling the slider buttons if the 'Tab' key is pressed, the focus goes to the handle and not to the button.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
    <ejs-slider id='buttons' :value='value' :showButtons="showButtons"
:type="type"></ejs-slider>
  </div>
</template>
<script setup>
import { SliderComponent as EjsSlider } from "@syncfusion/ej2-vue-inputs";
// Enable the button option in Slider
const showButtons = true;
const value = [30, 70];
const type = 'Range';
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    color: #008cff;
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
    <ejs-slider id='buttons' :value='value' :showButtons="showButtons"
:type="type"></ejs-slider>
  </div>
</template>
<script>
import { SliderComponent } from "@syncfusion/ej2-vue-inputs";
export default {
  components: {
    'ejs-slider': SliderComponent
  },
  data() {
    return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        // Enable the button option in Slider
        showButtons: true,
        value: [30, 70],
        type: 'Range'
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    color: #008cff;
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/getting-started-cs5" %}

See Also

Slider Formatting

Limits in Slider

Ticks in Slider

Getting Started with the Vue Range Slider Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Range Slider component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

Copyright © 2001 -2024 Syncfusion Inc.

or

`bash
yarn create vite
`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as `my-project` for this article.

`bash
? Project name: » my-project
`

2.Select `Vue` as the framework. It will create a Vue 3 project.

`bash
? Select a framework: » - Use arrow-keys. Return to submit.
Vanilla
Vue
React
Preact
Lit
Svelte
Others
`

3.Choose `JavaScript` as the framework variant to build this Vite project using JavaScript and Vue.

`bash
? Select a variant: » - Use arrow-keys. Return to submit.
JavaScript
TypeScript
Customize with create-vue ↗
Nuxt ↗
`

4.Upon completing the aforementioned steps to create the `my-project`, run the following command to install its dependencies:

`bash
cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Range Slider component as an example. To use the Vue Range Slider component in the project, the @syncfusion/ej2-vue-inputs package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Range Slider component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Add Syncfusion Vue component

Follow the below steps to add the Vue Range Slider component using Composition API or Options API:

1.First, import and register the Range Slider component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SliderComponent as EjsSlider} from "@syncfusion/ej2-vue-inputs";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SliderComponent} from "@syncfusion/ej2-vue-inputs";
//Component registration
export default {
name: "App",
components: {
'ejs-slider': SliderComponent
}
}
</script>
```

2.Add the component definition in template section.

~/SRC/APP.VUE

```
<template>
<ejs-slider id='default' :value='value'></ejs-slider>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-slider id='default' :value='value'></ejs-slider>
</div>
</template>
<script setup>
import { SliderComponent as EjsSlider} from "@syncfusion/ej2-vue-inputs";
const value = 30;
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-slider id='default' :value='value'></ejs-slider>
</div>
</template>
<script>
import { SliderComponent} from "@syncfusion/ej2-vue-inputs";
//Component registration
export default {
name: "App",
components: {
'ejs-slider': SliderComponent
},
data() {
return {
value: 30,
};
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

Types

The types of Range Slider are as follows:

| **Types** | **Usage** |

| --- | --- |

| Default | Shows a default Range Slider to select a single value. |

Copyright © 2001 -2024 Syncfusion Inc.

| MinRange | Displays the shadow from the start value to the current selected value. |

| Range | Selects a range of values. It also displays the shadow in-between the selection range. |

Both the Default Slider and Min-Range Slider have same behavior that is used to select a single value. In Min-Range Slider, a shadow is considered from the start value to current handle position. But the Range Slider contains two handles that is used to select a range of values and a shadow is considered in between the two handles.

In the following example demonstrates all types in Range Slider component.

COMPOSITION API (~/SRC/APP.VUE)

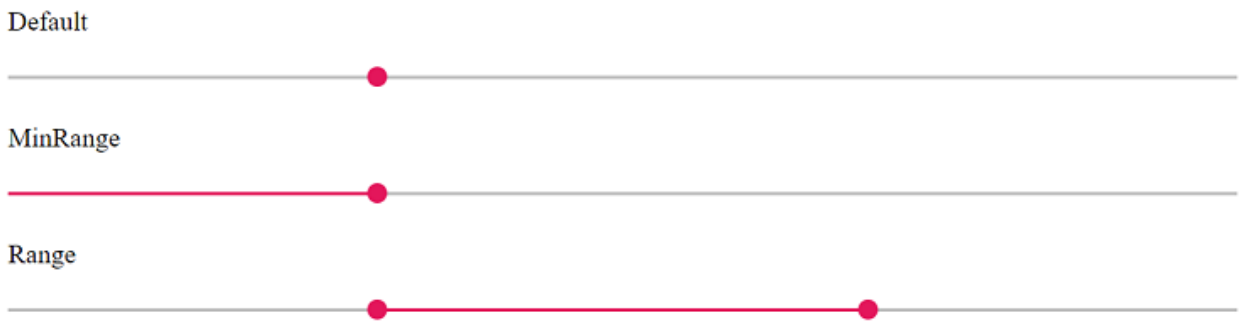
```
<template>
<div id="app">
<label class="labeltext">Default</label>
<ejs-slider id='default' :value='minvalue'></ejs-slider>
<label class="labeltext">MinRange</label>
<ejs-slider id='default' :value='minvalue' :type="mintype"></ejs-slider>
<label class="labeltext">Range</label>
<ejs-slider id='type' :value='value' :type="type"></ejs-slider>
</div>
</template>
<script setup>
import { SliderComponent as EjsSlider} from "@syncfusion/ej2-vue-inputs";
const minvalue = 30;
const value = [30, 70];
const type = 'Range';
const mintype = 'MinRange';
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<label class="labeltext">Default</label>
<ejs-slider id='default' :value='minvalue'></ejs-slider>
<label class="labeltext">MinRange</label>
<ejs-slider id='default' :value='minvalue' :type="mintype"></ejs-slider>
<label class="labeltext">Range</label>
<ejs-slider id='type' :value='value' :type="type"></ejs-slider>
</div>
</template>
<script>
import { SliderComponent} from "@syncfusion/ej2-vue-inputs";
//Component registration
export default {
name: "App",
components: {
'ejs-slider': SliderComponent
},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
data() {
return {
value: [30, 70],
minvalue: 30,
mintype: 'MinRange',
type: 'Range'
};
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```



Customization

Orientation

The Range Slider can be displayed, either in horizontal or vertical orientation. By default, the Range Slider renders in horizontal orientation.

In the following example, orientation property set as Vertical in Range Slider component.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-slider id='orientation' :value='value'
:orientation="orientation"></ejs-slider>
</div>
</template>
<script setup>
import { SliderComponent as EjsSlider} from "@syncfusion/ej2-vue-inputs";
const value = 30;
const orientation = 'Vertical';
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
#app {
color: #008cff;
height: 340px;
left: 30%;
position: absolute;
width: 50%;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-slider id='orientation' :value='value'
:orientation="orientation"></ejs-slider>
</div>
</template>
<script>
import { SliderComponent} from "@syncfusion/ej2-vue-inputs";
//Component registration
export default {
name: "App",
components: {
'ejs-slider': SliderComponent
},
data() {
return {
value: 30,
orientation: 'Vertical'
};
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#app {
color: #008cff;
height: 340px;
left: 30%;
position: absolute;
width: 50%;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Tooltip

The Range Slider displays the tooltip to indicate the current value by clicking the Range Slider bar or drag the Range Slider handle. The Tooltip position can be customized by using the placement property. Also decides the tooltip display mode on a page, i.e., on hovering, focusing, or clicking on the Range Slider handle. Tooltip always remains/displays on the page.

In the following example, tooltip property is enabled in Range Slider component.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-slider id='tooltip' :value='value' :tooltip="tooltip"
:type="type"></ejs-slider>
</div>
</template>
<script setup>
import { SliderComponent as EjsSlider} from "@syncfusion/ej2-vue-inputs";
const value = 30;
const type = 'MinRange';
const tooltip = { placement: 'After', isVisible: true, showOn: 'Always' };
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#app {
color: #008cff;
height: 40px;
left: 30%;
position: absolute;
top: 40%;
width: 50%;
```

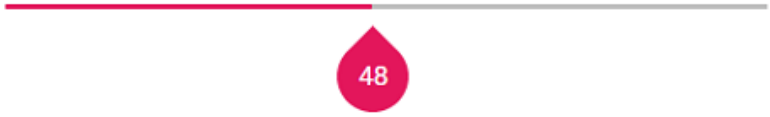
Copyright © 2001 -2024 Syncfusion Inc.

```
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-slider id='tooltip' :value='value' :tooltip="tooltip"
:type="type"></ejs-slider>
</div>
</template>
<script>
import { SliderComponent} from "@syncfusion/ej2-vue-inputs";
//Component registration
export default {
name: "App",
components: {
'ejs-slider': SliderComponent
},
data() {
return {
value: 30,
type: 'MinRange',
tooltip: { placement: 'After', isVisible: true, showOn: 'Always' }
};
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#app {
color: #008cff;
height: 40px;
left: 30%;
position: absolute;
top: 40%;
width: 50%;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.



Buttons

The Range Slider value can be changed by using the Increase and Decrease buttons. In Range Range Slider, by default the first handle value will be changed while clicking the button. Change the handle focus and press the button to change the last focused handle value.

After enabling the Range Slider buttons if the 'Tab' key is pressed, the focus goes to the handle and not to the button.

In the following example, showButtons property is enabled in Range Slider component.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-slider id='buttons' :value='value' :showButtons="showButtons"
:type="type"></ejs-slider>
</div>
</template>
<script setup>
import { SliderComponent as EjsSlider} from "@syncfusion/ej2-vue-inputs";
const value = [30, 70];
const type = 'Range';
const showButtons = true;
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#app {
color: #008cff;
height: 40px;
left: 30%;
position: absolute;
top: 40%;
width: 50%;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<div id="app">
<ejs-slider id='buttons' :value='value' :showButtons="showButtons"
:type="type"></ejs-slider>
</div>
</template>
<script>
import { SliderComponent} from "@syncfusion/ej2-vue-inputs";
//Component registration
export default {
name: "App",
components: {
'ejs-slider': SliderComponent
},
data() {
return {
value: [30, 70],
type: 'Range',
showButtons: true
};
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#app {
color: #008cff;
height: 40px;
left: 30%;
position: absolute;
top: 40%;
width: 50%;
}
</style>
```

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Copyright © 2001 -2024 Syncfusion Inc.

Two way binding in Vue Range slider component

It can be achieved by using the v-model directive in vue. In the following sample, when you change a value in one slider will automatically change the value in the other slider. It updates the other slider using value property.

APP.VUE

```
<template>
    <div id="app">
      <div>Default Slider</div>
    <ejs-slider id='default' v-model='value'></ejs-slider>
      <div>Min Range Slider</div>
    <ejs-slider id='minRange' type='MinRange' v-model='value'></ejs-slider>
  </div>
</template>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data() {
    return {
        value: 30,
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/two-way-binding-cs1" %}

Ticks in Vue Range slider component

The Ticks in Slider supports you to easily identify the current value/values of the Slider. It contains smallStep and largeStep. The value of the major ticks alone will be displayed in the slider. In order to enable/disable the small ticks, use the showSmallTicks property.

APP.VUE

```
<template>
    <div id="app">
    <ejs-slider id='ticks' :value='value' :tooltip="tooltip"
:ticks="ticks"></ejs-slider>
  </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data() {
    return {
       tooltip: { placement: 'Before', isVisible: true, showOn: 'Always' },
        value: 30,
        // Slider ticks customization
        ticks: { placement: 'After', largeStep: 20, smallStep: 10,
showSmallTicks: true }
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/ticks-cs2" %}

Step

When the Slider is moved, it increases/decreases the value based on the step value. By default, the value is increased/decreased by 1. Use the step property to change the increment step value.

APP.VUE

```
<template>
    <div id="app">
    <ejs-slider id='steps' :value='value' :tooltip="tooltip" :ticks="ticks"
:step="step"></ejs-slider>
  </div>
</template>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data() {
    return {
      ticks: { placement: 'After', largeStep: 20, smallStep: 10,
showSmallTicks: true },
        tooltip: { placement: 'Before', isVisible: true, showOn: 'Always' },
        value: 30,
```

```
        // Enables step
        step: 10
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/ticks-cs3" %}

Min and Max

Enables the minimum/starting and maximum/ending value of the Slider, by using the min and max property. By default, the minimum value is 1 and maximum value is 100. In the following sample the slider is rendered with the min value as 100 and max value as 1000.

APP.VUE

```
<template>
    <div id="app">
    <ejs-slider id='minmax' :value='value' :tooltip="tooltip" :ticks="ticks"
:min="min" :max="max"></ejs-slider>
  </div>
</template>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data() {
    return {
      ticks: { placement: 'After', largeStep: 200, smallStep: 100,
showSmallTicks: true },
        tooltip: { placement: 'Before', isVisible: true, showOn: 'Always' },
        // Minimum value
        min: 100,
        // Maximum value
        max: 1100,
        // Slider current value
        value: 400
    };
  }
}
</script>
<style>
```

```
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/ticks-cs4" %}

Format in Vue Range slider component

The format feature used to customize the units of Slider values to desired format. The formatted values will also be applied to the ARIA attributes of the slider. There are two ways of achieving formatting in slider.

- Use the format API of slider which utilizes our Internationalization to format values.
- Customize using the events namely renderingTicks and tooltipChange.

APP.VUE

```
<template>
    <div id="app">
    <ejs-slider id='format' :value='value' :min="min" :max="max"
:step="step"
    :tooltip="tooltip" :ticks="ticks" ></ejs-slider>
  </div>
</template>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data() {
    return {
      min: 0, max: 100, step: 1, value: 30,
    // Applying currency formatting for tooltip with two decimal specifiers
    tooltip: { isVisible: true, format: 'C2' },
    // Applying currency formatting for ticks with two decimal specifiers
    ticks: { placement: 'After', format: 'C2', largeStep: 20, smallStep: 10,
showSmallTicks: true }
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    #app {
     height: 40px;
     left: 30%;
     position: absolute;
     top: 40%;
     width: 50%;
   }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/format-cs2" %}

Using format API

In this method, we have different predefined formatting styles like Numeric (N), Percentage (P), Currency (C) and # specifiers. In this below example, we have formatted the ticks and tooltip values into percentage.

APP.VUE

```
<template>
    <div id="app">
    <ejs-slider id='api' :value='value' :min="min" :max="max" :step="step"
    :tooltip="tooltip" :ticks="ticks" ></ejs-slider>
  </div>
</template>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data() {
    return {
    min: 0, max: 1, value: 0.3, step: 0.01,
    // Assigning ticks values to percentage formatting
    ticks: { placement: 'After', largeStep: .2, smallStep: .1,
showSmallTicks: true, format: 'P0' },
    // Assigning tooltip values to percentage formatting
    tooltip: { placement: 'Before', isVisible: true, showOn: 'Always',
format: 'P0' }
  }
}
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/range-slider/format-cs3" %}

Using Events

In this method, we will be retrieving the values from the slider events then process them to desired formatted the values. In this sample we have customized the ticks values into weekdays as one formatting and tooltip values into day of the week as another formatting.

APP.VUE

```
<template>
    <div id="app">
    <ejs-slider id='events' :value='value' :tooltip="tooltip" :ticks="ticks"
:min="min" :max="max" :renderingTicks="renderingTicks"
:tooltipChange="tooltipChange"></ejs-slider>
  </div>
</template>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data() {
    return {
       // Minimum value
        min: 0,
        // Maximum value
        max: 6,
        // Slider current value
        value: 2,
        // Assigning ticks data
        ticks: {
            placement: 'After',
            largeStep: 1
        },
        // Assigning tooltip data
        tooltip: {
            placement: 'Before',
            isVisible: true
        }
    };
  },
  methods: {
       renderingTicks: function (args) {
                // Weekdays Array
                var daysArr =
['Sunday','Monday','Tuesday','Wednesday','Thrusday','Friday','Saturday'];
                // Customizing each ticks text into weeksdays
                args.text = daysArr[parseFloat(args.value)];
        },
        tooltipChange: function (args) {
                // Customizing tooltip to display the Day (in numeric) of
the week
                args.text =  'Day ' + (Number(args.value) + 1).toString();
        }
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/format-cs4" %}

Limits in Vue Range slider component

The slider limits restrict the slider thumb between a particular range. This is used if higher or lower value affects the process or product where the slider is being used.

The following are the six options in the slider's limits object. Each API in the limits object is optional.

- enabled: Enables the limits in the Slider.
- minStart: Sets the minimum limit for the first handle.
- minEnd: Sets the maximum limit for the first handle.
- maxStart: Sets the minimum limit for the second handle.
- maxEnd: Sets the maximum limit for the second handle.
- startHandleFixed: Locks the first handle.
- endHandleFixed: Locks the second handle.

Default and MinRange Slider limits

There is only one handle in the Default and MinRange Slider, so minStart, minEnd, and startHandleFixed options can be used. When the limits are enabled in the Slider, the limited area becomes darken. So you can differentiate the allowed and restricted area. Refer to the following snippet to enable the limits in the Slider.

`ts

.....

limits: { enabled: true, minStart: 10, minEnd: 40 }

.....

`

APP.VUE

```
<template>
    <div id="app">
    <ejs-slider id='default' :value='value' :limits='limits'
:tooltip='tooltip' :type='type'></ejs-slider>
  </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data() {
    return {
        value: 30,
        tooltip: { isVisible: true },
        limits: { enabled: true, minStart: 10, minEnd: 40 },
        type: 'MinRange'
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";;
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/default-limit-cs1" %}

Range Slider limits

In the range slider, both handles can be restricted and locked from the limit's object. In this sample, the first handle is limited between 10 and 40, and the second handle is limited between 60 and 90.

`ts

.....

limits: { enabled: true, minStart: 10, minEnd: 40, maxStart: 60, maxEnd: 90 }

.....

`

APP.VUE

```
<template>
    <div id="app">
    <ejs-slider id='default' :value='value' :limits='limits'
:tooltip='tooltip' :type='type'></ejs-slider>
  </div>
</template>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    return {
        value: [30, 70],
        tooltip: { isVisible: true },
        limits: { enabled: true, minStart: 10, minEnd: 40, maxStart: 60,
maxEnd: 90 },
        type: 'Range'
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/range-limit-cs1" %}

Handle lock

The movement of slider handles can be locked by enabling the startHandleFixed and endHandleFixed properties in the limit's object.

In this sample, the movement of both slider handles has been locked.

`ts

.....

limits: { enabled: true, startHandleFixed: true, endHandleFixed: true }

.....

`

APP.VUE

```
<template>
    <div id="app">
    <ejs-slider id='default' :value='value' :limits='limits'
:tooltip='tooltip' :type='type'></ejs-slider>
  </div>
</template>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data() {
    return {
        value: [30, 70],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        tooltip: { isVisible: true },
        limits: { enabled: true, startHandleFixed: true, endHandleFixed:
true },
        type: 'Range'
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    height: 40px;
    left: 30%;
    position: absolute;
    top: 40%;
    width: 50%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/handle-lock-cs1" %}

Style in Vue Range slider component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the slider track

Use the following CSS to customize the slider track.

`

.e-control-wrapper.e-slider-container.e-horizontal .e-slider-track {

background: #007bff;

height: 3px;

}

`

Customizing the slider handle

Use the following CSS to customize the slider handle properties.

`

.e-control-wrapper.e-slider-container .e-slider .e-handle {

background-color: #f9920b;

border-radius: 50%;

border: 0;

}

`

Copyright © 2001 -2024 Syncfusion Inc.

Customizing the slider limits

Use the following CSS to customize the slider limits.

```
.e-control-wrapper.e-slider-container.e-horizontal .e-limits {
background-color: rgba(69, 100, 233, 0.46);
}
```

Customizing the slider ticks

Use the following CSS to customize the slider ticks.

```
.e-scale .e-tick.e-custom::before {
content: '\e967';
position: absolute;
}
```

Customizing the slider buttons

Use the following CSS to customize the slider buttons.

```
.e-control-wrapper.e-slider-container .e-slider-button {
background: #007bff;
height: 25px;
width: 25px;
}
```

Accessibility in Vue Range Slider component

The Range Slider component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Range Slider component is outlined below.

Accessibility Criteria	Compatibility
WCAG 2.2 Support	
Section 508 Support	

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Range Slider component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Range Slider component:

| Attributes | Purpose |

| --- | --- |

| `role=slider` | Used to convey a significant and contextual message to the user. |

| `aria-valuemin` | Indicates the Minimum value of the slider. |

| `aria-valuemax` | Indicates the Maximum value of the slider. |

| `aria-valuenow` | Indicates the current value of the slider. |

| `aria-valuetext` | Returns the current text of the slider. |

Copyright © 2001 -2024 Syncfusion Inc.

| aria-orientation | Indicates whether the Slider is oriented horizontally or vertically. |

| aria-label | Provides an accessible name for the Slider, serving as label text for the Slider's left and right buttons (for increment and decrement). |

Keyboard interaction

The Range Slider component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Range Slider component.

| **Press** | **To do this** |

| --- | --- |

| Right Arrow/Up Arrow | Increase the Slider value. |

| Left Arrow/Down Arrow | Decrease the Slider value. |

| Home | Moves to the start value (for Range Slider when the second thumb is focused and the Home key is pressed, it moves to the first thumb value). |

| <kbd>End | Moves to the end value (for Range Slider when the first thumb is focused and the End key is pressed, it moves to the second thumb value). |

| Page Up | Increases the Slider by largeStep value. |

| Page Down | Decreases the Slider by largeStep value. |

Ensuring accessibility

The Range Slider component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Range Slider component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Range Slider component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/slider.html" %}

See also

- Accessibility in Syncfusion Vue components

How To

Date range slider in Vue Range slider component

The date formatting can be achieved in ticks and tooltip using renderingTicks and tooltipChange events, respectively. The process of formatting is explained in the following sample.

APP.VUE

```
<template>
  <div id="app">
    <div class="wrap">
      <ejs-slider
        id="ticks"
        :min="minVal"
        :max="maxVal"
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        :value="value"
        :tooltip="tooltip"
        :ticks="ticks"
        showButtons="true"
        :step="stepVal"
        v-on:renderingTicks="onRenderingTicks"
        v-on:tooltipChange="onTooltipChange"
      ></ejs-slider>
    </div>
  </div>
</template>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
#app {
  height: 40px;
  position: absolute;
  width: 98%;
}
.wrap {
  box-sizing: border-box;
  height: 260px;
  margin: 0 auto;
  padding: 30px 10px;
  width: 460px;
}
</style>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data: function() {
    return {
      tooltip: { placement: "Before", isVisible: true },
      minVal: new Date("2013-06-13").getTime(),
      maxVal: new Date("2013-06-21").getTime(),
      value: new Date("2013-06-15").getTime(),
      stepVal: 86400000,
      // Slider ticks customization
      ticks: { placement: "After", largeStep: 2 * 86400000 }
    };
  },
  methods: {
    onTooltipChange(args) {
      let totalMiliSeconds = Number(args.text);
      // Convert the current milliseconds to the respective date in desired
format
      let custom = { year: "numeric", month: "short", day: "numeric" };
      args.text = new Date(totalMiliSeconds).toLocaleDateString(
        "en-us",
        custom
      );
    },
    onRenderingTicks(args) {
```

```
      let totalMiliSeconds = Number(args.value);
      // Convert the current milliseconds to the respective date in desired
format
      let custom = { year: "numeric", month: "short", day: "numeric" };
      args.text = new Date(totalMiliSeconds).toLocaleDateString(
        "en-us",
        custom
      );
    }
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/range-slider/how-to-cs1" %}

Time range slider in Vue Range slider component

The time formatting can be achieved in the same manner as date formatting using renderingTicks and change events. The process of time formatting is explained in the following sample.

APP.VUE

```
<template>
  <div id="app">
    <div class="wrap">
      <ejs-slider
        id="ticks"
        :min="minVal"
        :max="maxVal"
        :value="value"
        :tooltip="tooltip"
        :ticks="ticks"
        showButtons="true"
        :step="stepVal"
        v-on:renderingTicks="onRenderingTicks"
        v-on:tooltipChange="onTooltipChange"
      ></ejs-slider>
    </div>
  </div>
</template>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#app {
  height: 40px;
  position: absolute;
  width: 98%;
}
.wrap {
  box-sizing: border-box;
  height: 260px;
  margin: 0 auto;
  padding: 30px 10px;
  width: 460px;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data: function() {
    return {
      tooltip: { placement: "Before", isVisible: true },
      minVal: new Date(2013, 6, 13, 11).getTime(),
      maxVal: new Date(2013, 6, 13, 17).getTime(),
      value: new Date(2013, 6, 13, 13).getTime(),
      stepVal: 3600000,
      // Slider ticks customization
      ticks: { placement: "After", largeStep: 2 * 3600000 }
    };
  },
  methods: {
    onTooltipChange(args) {
      let totalMiliSeconds = Number(args.text);
      let custom = { hour: "2-digit", minute: "2-digit" };
      args.text = new Date(totalMiliSeconds).toLocaleTimeString(
        "en-us",
        custom
      );
    },
    onRenderingTicks(args) {
      let totalMiliSeconds = Number(args.value);
      let custom = { hour: "2-digit", minute: "2-digit" };
      args.text = new Date(totalMiliSeconds).toLocaleTimeString(
        "en-us",
        custom
      );
    }
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/range-slider/time-format-cs1" %}

Numeric range slider in Vue Range slider component

The numeric values can be formatted into different decimal digits or fixed number of whole numbers or to represent units. The numeric processing is demonstrated as follows.

APP.VUE

```
<template>
   <div id='app'>
     <div class='wrap'>
            <div class='label'>Slider formatted with unit
representation</div>
            <ejs-slider id="slider"  min=0 max=100 step=1 value=30
:ticks='ticks' :tooltip='tooltip'>
            </ejs-slider>
        </div>
        <div class='wrap'>
```

```
            <div class='label'>Slider formatted with three decimal
specifiers</div>
            <ejs-slider id="slider1" min=0.1 max=0.2 step=0.01 value=0.13
:tooltip='decimalTooltip' :ticks='decimalTicks'>
            </ejs-slider>
        </div>
        <div class='wrap'>
            <div class='label'>Slider formatted with two leading zeros</div>
            <ejs-slider id="slider2" min=0 max=100 step=1 value=30
:tooltip='formatTooltip' :ticks='formatTicks'>
            </ejs-slider>
        </div>
   </div>
</template>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #app {
        height: 40px;
        position: absolute;
        width: 98%;
    }
    .wrap {
    box-sizing: border-box;
    height: 100px;
    margin: 0 auto;
    padding: 30px 10px;
    width: 460px;
    }
    .wrap .label {
    text-align: center;
    }
</style>
<script>
import Vue from 'vue';
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data: function() {
    return {
        tooltip: { isVisible: true, format: '##.## Km' },
        ticks: { placement: 'After', format: '##.## Km', largeStep: 20,
smallStep: 10, showSmallTicks: true },
        decimalTooltip:{ isVisible: true, format: '##.#00' },
        decimalTicks:{ placement: 'After', format: '##.#00', largeStep:
0.02, smallStep: 0.01, showSmallTicks: true },
        formatTooltip:{ isVisible: true, format: '00##' },
        formatTicks:{ placement: 'After', format: '00##', largeStep: 20,
smallStep: 10, showSmallTicks: true }
    };
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/range-slider/numeric-cs1" %}

Customize slider bar in Vue Range slider component

Slider appearance can be customized through CSS. By overriding the slider CSS classes, you can customize the slider bar. The slider bar can be customized with different themes. By default, slider have class name e-slider-track for bar. The class can be overridden with our own color values like the following code snippet.

```
.e-control.e-slider .e-slider-track .e-range {
background: linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%, #2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
}
```

You can also apply background color for a certain range depending upon slider values, using change event.

```ts
onChange(args){
let sliderTrack = document.getElementById('dynamic*color*slider').querySelector('.e-range');
let sliderHandle = document.getElementById('dynamic*color*slider').querySelector('.e-handle');
if (args.value > 0 && args.value <= 25) {
// Change handle and range bar color to green when
(sliderHandle).style.backgroundColor = 'green';
(sliderTrack).style.backgroundColor = 'green';
} else if (args.value > 25 && args.value <= 50) {
// Change handle and range bar color to royal blue
(sliderHandle).style.backgroundColor = 'royalblue';
(sliderTrack).style.backgroundColor = 'royalblue';
} else if (args.value > 50 && args.value <= 75) {
// Change handle and range bar color to dark orange
(sliderHandle).style.backgroundColor = 'darkorange';
(sliderTrack).style.backgroundColor = 'darkorange';
} else if (args.value > 75 && args.value <= 100) {
// Change handle and range bar color to red
(sliderHandle).style.backgroundColor = 'red';
(sliderTrack).style.backgroundColor = 'red';
}
```

Copyright © 2001 -2024 Syncfusion Inc.

}
`

APP.VUE

```
<template>
  <div id="app">
    <div class="col-lg-12 control-section">
      <div class="slider-content-wrapper">
        <div class="slider_container">
          <div class="slider-labeltext slider_userselect">Height</div>
          <!-- Square slider element -->
          <ejs-slider id="height_slider" value="30" min="0" max="100"></ejs-
slider>
        </div>
        <div class="slider_container">
          <div class="slider-labeltext slider_userselect">Gradient
color</div>
          <ejs-slider id="gradient_slider" min="0" max="100" value="30"
type="MinRange"></ejs-slider>
        </div>
        <div class="slider_container">
          <div class="slider-labeltext slider_userselect">Dynamic thumb and
selection bar color</div>
          <ejs-slider
            id="dynamic_color_slider"
            min="0"
            max="100"
            value="30"
            type="MinRange"
            v-on:created="onCreated"
            v-on:change="onChange"
          ></ejs-slider>
        </div>
      </div>
    </div>
  </div>
</template>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#app {
  height: 40px;
  position: absolute;
  width: 98%;
}
.slider-content-wrapper {
  width: 40%;
  margin: 0 auto;
  min-width: 185px;
}
.slider-userselect {
  -webkit-user-select: none;
  /* Safari 3.1+ */
```

```
  -moz-user-select: none;
  /* Firefox 2+ */
  -ms-user-select: none;
  /* IE 10+ */
  user-select: none;
  /* Standard syntax */
}
.slider-labeltext {
  text-align: left;
  font-weight: 500;
  font-size: 13px;
  padding-bottom: 10px;
}
#height_slider .e-handle,
#gradient_slider .e-handle {
  height: 20px;
  width: 20px;
  margin-left: -10px;
  top: calc(50% - 10px);
}
.slider_container {
  margin-top: 40px;
}
#height_slider .e-tab-handle::after {
  background-color: #f9920b;
}
#height_slider .e-slider-track {
  height: 8px;
  top: calc(50% - 4px);
  border-radius: 0;
}
#gradient_slider .e-range {
  height: 6px;
  top: calc(50% - 3px);
  border-radius: 5px;
  background: -webkit-gradient(
    linear,
    left top,
    left bottom,
    color-stop(0%, #1e5799),
    color-stop(100%, #7db9e8)
  );
  background: -webkit-linear-gradient(
    left,
    #e1451d 0,
    #fdff47 17%,
    #86f9fe 50%,
    #2900f8 65%,
    #6e00f8 74%,
    #e33df9 83%,
    #e14423 100%
  );
  background: linear-gradient(
    left,
    #e1451d 0,
    #fdff47 17%,
    #86f9fe 50%,
```

```
    #2900f8 65%,
    #6e00f8 74%,
    #e33df9 83%,
    #e14423 100%
  );
  background: -moz-linear-gradient(
    left,
    #e1451d 0,
    #fdff47 17%,
    #86f9fe 50%,
    #2900f8 65%,
    #6e00f8 74%,
    #e33df9 83%,
    #e14423 100%
  );
}
#gradient_slider .e-slider-track {
  height: 8px;
  top: calc(50% - 4px);
  border-radius: 5px;
}
</style>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
import { enableRipple } from "@syncfusion/ej2-base";
enableRipple(true);
export default {
  data: function() {
    return {};
  },
  methods: {
    onCreated(args) {
      let sliderTrack = document
        .getElementById("dynamic_color_slider")
        .querySelector(".e-range");
      let sliderHandle = document
        .getElementById("dynamic_color_slider")
        .querySelector(".e-handle");
      sliderHandle.style.backgroundColor = "green";
      sliderTrack.style.backgroundColor = "green";
    },
    onChange(args) {
      let sliderTrack = document
        .getElementById("dynamic_color_slider")
        .querySelector(".e-range");
      let sliderHandle = document
        .getElementById("dynamic_color_slider")
        .querySelector(".e-handle");
      if (args.value > 0 && args.value <= 25) {
        // Change handle and range bar color to green when
        sliderHandle.style.backgroundColor = "green";
        sliderTrack.style.backgroundColor = "green";
      } else if (args.value > 25 && args.value <= 50) {
        // Change handle and range bar color to royal blue
        sliderHandle.style.backgroundColor = "royalblue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            sliderTrack.style.backgroundColor = "royalblue";
        } else if (args.value > 50 && args.value <= 75) {
            // Change handle and range bar color to dark orange
            sliderHandle.style.backgroundColor = "darkorange";
            sliderTrack.style.backgroundColor = "darkorange";
        } else if (args.value > 75 && args.value <= 100) {
            // Change handle and range bar color to red
            sliderHandle.style.backgroundColor = "red";
            sliderTrack.style.backgroundColor = "red";
        }
      }
    }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/range-slider/bar-cs1" %}

Customize slider limits in Vue Range slider component

Slider appearance can be customized via CSS. By overriding the slider CSS classes, the slider limit bar can be customized. Here, the limit bar is customized with different background color. By default, the slider has class e-limits for limits bar. You can override the class with our own color values as given in the following code snippet.

`

.e-slider-container.e-horizontal .e-limits {

background-color: rgba(69, 100, 233, 0.46);

}

`

APP.VUE

```
<template>
   <div id='app'>
      <div class="col-lg-8 control-section">
            <div class="content-wrapper">
                <div class="sliderwrap">
                    <label class="userselect">MinRange Slider With
Limits</label>
                    <ejs-slider id="minrange" min=0 max=100 value=30  step=1
type='MinRange' :ticks='ticks' :limits='limits' :tooltip='tooltip'></ejs-
slider>
                </div>
                <div class="sliderwrap">
                    <label class="userselect">Range Slider With
Limits</label>
                    <ejs-slider id="range" min=0 max=100 :value='value'
step=1 type='Range' :ticks='rangeTicks' :tooltip='rangeTooltip'
:limits='rangeLimits'></ejs-slider>
                </div>
            </div>
        </div>
   </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #app {
        height: 40px;
        position: absolute;
        width: 98%;
    }
    .content-wrapper {
    width: 52%;
    margin: 0 auto;
    min-width: 185px;
    }
    .sliderwrap {
    margin-top: 45px;
    }
    .e-bigger .content-wrapper {
    width: 80%;
    }
    .sliderwrap label {
    padding-bottom: 50px;
    font-size: 13px;
    font-weight: 500;
    margin-top: 15px;
    display: block;
    }
    .userselect {
    -webkit-user-select: none;
    /* Safari 3.1+ */
    -moz-user-select: none;
    /* Firefox 2+ */
    -ms-user-select: none;
    /* IE 10+ */
    user-select: none;
    /* Standard syntax */
    }
    .property-custom td {
    padding: 5px;
    }
    #range .e-limits, #minrange .e-limits {
    background-color: #ccc;
    background-color: rgba(69, 100, 233, 0.46);
    }
</style>
<script>
import Vue from 'vue';
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
  data: function() {
    return {
            ticks:{ placement: 'Before', largeStep: 20, smallStep: 5,
showSmallTicks: true },
```

```
            limits: { enabled: true, minStart: 10, minEnd: 40 },
            tooltip:{ isVisible: true, placement: 'Before', showOn: 'Focus'
},
            rangeTicks: { placement: 'Before', largeStep: 20, smallStep: 5,
showSmallTicks: true },
            rangeLimits: { enabled: true, minStart: 10, minEnd: 40,
maxStart: 60, maxEnd: 90 },
            rangeTooltip:{ isVisible: true, placement: 'Before', showOn:
'Focus' },
            value:[30,70]
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/range-slider/limits-cs1" %}

Customize slider ticks label in Vue Range slider component

Slider view can be customized via CSS. By overriding the slider CSS classes, you can customize the ticks. The ticks in slider allows you to easily identify the current value/values of the slider. It contains smallStep and largeStep. By default, slider has class e-tick for slider ticks. You can override the class as per your requirement. Refer to the following code snippet to render ticks.

`ts

.e-scale .e-tick.e-custom::before {

content: '\e967';

position: absolute;

}

`

Here, the color for rendered ticks has been applied through nth-child(child*number). The color is applied to the value of the child*number in the slider.

`ts

ticks_slider .e-scale :nth-child(1)::before {
color: red;

}

`

APP.VUE

```
<template>
  <div id="app">
    <div id="container">
      <div class="col-lg-12 control-section">
        <div class="slider-content-wrapper">
          <div class="slider_container" id="slider_wrapper">
            <div class="slider_labelText userselect">Dynamic ticks
color</div>
            <!-- Ticks slider element -->
```

```
            <ejs-slider
              id="ticks_slider"
              min="0"
              max="100"
              value="30"
              step="5"
              type="MinRange"
              :ticks="ticks"
              v-on:renderingTicks="onRenderingTicks"
            ></ejs-slider>
          </div>
          <div class="slider_container">
            <div class="slider_labelText userselect">Ticks with
legends</div>
            <!-- Ticks slider element -->
            <ejs-slider
              id="slider"
              min="0"
              max="100"
              value="30"
              type="MinRange"
              :ticks="rangeTicks"
              v-on:renderedTicks="onRangeTicks"
            ></ejs-slider>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#app {
  height: 40px;
  position: absolute;
  width: 98%;
}
.slider-content-wrapper {
  width: 40%;
  margin: 0 auto;
  min-width: 185px;
}
.userselect {
  -webkit-user-select: none;
  /* Safari 3.1+ */
  -moz-user-select: none;
  /* Firefox 2+ */
  -ms-user-select: none;
  /* IE 10+ */
  user-select: none;
  /* Standard syntax */
}
.slider_labelText {
  text-align: left;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  font-weight: 500;
  font-size: 13px;
  padding-bottom: 40px;
}
.slider_container {
  margin-top: 40px;
}
.e-bigger .slider-content-wrapper {
  width: 80%;
}
#ticks_slider .e-range {
  z-index: unset;
}
#ticks_slider .e-scale .e-tick {
  background-image: none;
  visibility: visible;
  font-family: "e-customized-icons";
}
@font-face {
  font-family: "e-customized-icons";
  src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj8iS4cAAAEoAAAAVmNtYXDS5tJrAAABjAAAAEBnbHlmdMA
KbQAAAdQAAAOwaGVhZBNseyYAAADQAAAANmhoZWEHogNjAAArAAAACRobXR4C9AAAAAAAYAAAAA
MbG9jYQCaAdgAAAHMAAAACG1heHABEAEuAAABCAAAACBuYW1lc0cOBgAABYQAAAIlcG9zdNSlKbQ
AAAesAAAARwABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAAAAAAwABAAAAAQAAtxzLE18
PPPUACwPoAAAAANgtmycAAAAA2C2bJwAAAAAD8wPzAAAACAACAAAAAAAAAAAEAAAADASIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPwAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA6QLpZwNS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAAAAAAgAAAAMAAAAUAAMAAQAAABQABAAsAAAABgAEAAEAAukC6Wf//wAA6QLpZ///AAAAAAA
BAAYABgAAAAEAAgAAAAAAmgHYAAIAAAAAA+oD6gAzAIcAAAEzHxghNT8WEx8THQEPEisBLxI9AT8
SAgAQECQmKCgpKScTEhIREA8ODQwKCgQHBQQBAfwqAQMFBgcKCgwNDg8QERISEycpKSgoJiQgDQw
MDAwXFhUUEhEPDQsJCAIDAQEBAQMCCAkLDQ8REhQVFhcMDAwMDQ0MDAwMFxYVFBIRDw0LCQgCAwE
BAQEDAggJCw0PERIUFRYXDAwMDAGFAQMEBwkKDQ4ICAkKCgoLCwwMDAcNDg8Og3sPDw4NDgwMDAs
LCgoKCQgIDg0KCQcEAwJnAQEBAgMHCgsNDxESExUWFwwMDQwNDA0MDAwXFhUTExAPDQwJBwMCAgE
BAgIDBwkMDQ8QExMVFhcMDAwNDA0MDQwMFxYVExIRDw0LCgcDAgEBAAAAAwAAAAD8wPzAF8AwAE
hAAABDxMfFz8XLxcPAjcfFA8XLxc/Fx8CJw8UHxc/Fy8XDwIBqRQUFBISERAQDg0NCwoJBwcFBAI
BAQIEBQcHCQoLDQ0OEBAREhIUFBQVFhYWFhYWFRUTFBISERAQDg0NCwoJBwcFBAIBAQIEBQcHCQo
LDQ0OEBAREhIUExUVFhYWFhYWtg4NGxkZGBYWFRMSEA8OCwsIBwUDAQEDBQcICwsODxASExUWFhg
ZGRsbHB0dHh4dHRwbGxkZGBYWFRMSEA8NDAsIBwUDAQEDBQcICwsODxASExUVFxgZGRsbHB0dHh4
dHd0QDx4eHBsaGRcWFRIREA0MCQgGAwEBAwYICQwNEBESFRYXGRobHB4eHyEgIiIiIiAhHx4eHBs
aGRcWFRIREA0MCQgGAwEBAwYICQwNEBESFRYXGRobHB4eHyEgIiIiIiEDPAYICQoLDQ0OEBAREhI
TFBUVFRYXFhYWFRQUFBISERAQDg0MDAoJBwcFBAIBAQIEBQcHCQoMDA0OEBAREhIUFBQVFhYWFxY
VFRUUExISERAQDg0NCwoJCAYFBAIBAQIEZAQECgwODxASExUVFxgYGhsbHB0dHh4dHRwbGxkZGBY
WFBQSEA8NDAoJBwUDAQEDBQcICwsODxASExUWFhgZGRsbHB0dHh4dHRwbGxoYGBcVFRMSEA8OCws
IBwUDAQEDBTYFBQwNEBESFRYXGRobHB0fHyEgIiIiIiEgHx4eHBsaGRcWFBMRDw4MCQgGAwEBAwY
ICQwODxETFBYXGRobHB4eHyEgIiIiIiAhHx4eHBsaGRcWFRIRDw4MCQgGAwEBAwYAAAAAAAASAN4
AAQAAAAAAAAABAAAAAQAAAAAAQAHAAEAAQAAAAAAgAHAAgAAQAAAAAAAwAHAA8AAQAAAAAABAA
HABYAAQAAAAAABQALAB0AAQAAAAAABgAHACgAAQAAAAAACgAsAC8AAQAAAAAACwASAFsAAwABBAk
AAAACAG0AAwABBAkAAQAOAG8AAwABBAkAAgAOAH0AAwABBAkAAwAOAIsAAwABBAkABAAOAJkAAwA
BBAkABQAWAKcAAwABBAkABgAOAL0AAwABBAkACgBYAMsAAwABBAkACwAkASMgZS1pY29uc1JlZ3V
sYXJlLWljb25zZS1pY29uc1ZlcnNpb24gMS4wZS1pY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN
5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGkAYwBvAG4AcwB
SAGUAZwB1AGwAYQByAGUALQBpAGMAbwBuAHMAZQAtAGkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACA
AMQAuADAAZQAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwBlAG4AZQByAGEAdABlAGQAIAB1AHMAaQB
uAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHc
ALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAAAAAAKAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAADAQIBAwEEAAh0ZW1wLWN1cxJGQl9DaGVja2JveF9zZWxlY3QAAAA=)
```

```
    format("truetype");
  font-weight: normal;
  font-style: normal;
}
#ticks_slider .e-scale {
  z-index: 0;
}
#ticks_slider .e-scale .e-custom::before {
  content: "\e967";
  position: absolute;
}
#ticks_slider .e-scale :nth-child(1)::before {
  color: red;
}
#ticks_slider .e-scale :nth-child(2)::before {
  color: blue;
}
#ticks_slider .e-scale :nth-child(3)::before {
  color: green;
}
#ticks_slider .e-scale :nth-child(4)::before {
  color: blueviolet;
}
#ticks_slider .e-scale :nth-child(5)::before {
  color: orange;
}
#ticks_slider .e-scale :nth-child(6)::before {
  color: pink;
}
#ticks_slider .e-scale .e-custom::before {
  font-size: 10px;
}
#ticks_slider .e-scale .e-custom::before {
  top: calc(50% + 1px);
  left: calc(50% - 5px);
}
#slider_wrapper #ticks_slider .e-scale :nth-child(1)::before {
  top: calc(50% + 1px);
  left: calc(0% - 5px);
}
#slider_wrapper #ticks_slider .e-scale :nth-child(6)::before {
  top: calc(50% + 1px);
  left: calc(100% - 6px);
}
</style>
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
import { enableRipple } from "@syncfusion/ej2-base";
enableRipple(true);
export default {
  data: function() {
    return {
      ticks: { placement: "Before", largeStep: 20 },
      rangeTicks: { placement: "Both", largeStep: 20, smallStep: 5 }
    };
```

```
  },
  methods: {
    onRenderingTicks(args) {
      if (args.tickElement.classList.contains("e-large")) {
        args.tickElement.classList.add("e-custom");
      }
    },
    onRangeTicks(args) {
      let li = args.ticksWrapper.getElementsByClassName("e-large");
      let remarks = [
        "Very Poor",
        "Poor",
        "Average",
        "Good",
        "Very Good",
        "Excellent"
      ];
      for (let i = 0; i < li.length; ++i) {
        li[i].querySelectorAll(".e-tick-both")[1].innerText = remarks[i];
      }
    }
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/range-slider/ticks-cs1" %}

Customize slider thumb in Vue Range slider component

Slider appearance can be customized through CSS. By overriding the slider CSS classes, you can customize the thumb. By default, slider has unique class e-handle for slider thumb. You can override the following class as per your requirement. Here, in the sample, the slider thumb has been customized to square, circle, oval shapes, and background image has also been customized.

`ts

.e-control.e-slider .e-handle {

background-image: url('https://ej2.syncfusion.com/demos/src/slider/images/thumb.png');

background-color: transparent;

height: 25px;

width: 25px;

}

square_slider.e-control.e-slider .e-handle {

border-radius: 0%;

background-color: #f9920b;

border: 0;

}

circle_slider.e-control.e-slider .e-handle {

border-radius: 50%;

Copyright © 2001 -2024 Syncfusion Inc.

background-color: #f9920b;

border: 0;

}

oval_slider.e-control.e-slider .e-handle {
height: 25px;

width: 8px;

top: 3px;

border-radius: 15px;

background-color: #f9920b;

}

`

APP.VUE

```
<template>
   <div id='app'>
      <div class="col-lg-12 control-section">
            <div class="slider-content-wrapper">
                <div class="slider_container">
                    <div class="labelText slider-userselect">Square</div>
                    <!-- Square slider element -->
                    <ejs-slider id="square_slider" value=30 min=0
max=100></ejs-slider>
                </div>
                <div class="slider_container">
                    <div class="labelText slider-userselect">Circle</div>
                    <!-- Circle slider element -->
                    <ejs-slider id="circle_slider" min=0 max=100
value=30></ejs-slider>
                </div>
                <div class="slider_container">
                    <div class="labelText slider-userselect">Oval</div>
                    <!-- Oval slider element -->
                    <ejs-slider id="oval_slider" value=30 min=0
max=100></ejs-slider>
                </div>
                <div class="slider_container">
                    <div class="labelText slider-userselect">Custom
image</div>
                    <!-- Image slider element -->
                    <ejs-slider id="image_slider" value=30 min=0 max=100
:ticks='ticks'></ejs-slider>
                </div>
            </div>
        </div>
   </div>
</template>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #app {
        height: 40px;
        position: absolute;
        width: 98%;
    }
    .slider-content-wrapper {
    width: 40%;
    margin: 0 auto;
    min-width: 185px;
    }
    .slider-userselect {
    -webkit-user-select: none;
    /* Safari 3.1+ */
    -moz-user-select: none;
    /* Firefox 2+ */
    -ms-user-select: none;
    /* IE 10+ */
    user-select: none;
    /* Standard syntax */
    }
    .labelText {
    text-align: left;
    font-weight: 500;
    font-size: 13px;
    padding-bottom: 10px;
    }
    .slider_container {
    margin-top: 40px;
    }
    .e-bigger .content-wrapper {
    width: 80%;
    }
    #square_slider .e-handle {
    border-radius: 0;
    background-color: #f9920b;
    border: 0;
    }
    #circle_slider .e-handle {
    background-color: #f9920b;
    border-radius: 50%;
    border: 0;
    }
    #image_slider .e-handle {
    height: 25px;
    width: 24px;
    background-size: 24px;
    }
    #image_slider .e-handle {
    background-image:
url('https://ej2.syncfusion.com/demos/src/slider/images/thumb.png');
    background-repeat: no-repeat;
    background-color: transparent;
    border: 0;
    }
    #square_slider .e-tab-handle::after,
```

```
    #circle_slider .e-tab-handle::after {
    background-color: #f9920b;
    }
    #image_slider .e-tab-handle::after {
    background-color: transparent;
    }
    #oval_slider .e-handle {
    height: 25px;
    width: 8px;
    top: 3px;
    border-radius: 15px;
    background-color: #f9920b;
    }
</style>
<script>
import Vue from 'vue';
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
  data: function() {
   return {
            ticks:{ placement: 'After'}
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/range-slider/thumb-cs1" %}

Form slider with form validator in Vue Range slider component

The Slider component can be validated using our FormValidator. The following steps walk-through slider validation.

- Render slider component inside a form.
- Bind [changed]../api/slider#changed) event in the slider component to validate the slider value when the value changes.
- Initialize and render FormValidator for the form using form ID.

`ts

// Initialize Form validation

let formObj: FormValidator;

formObj = new FormValidator("#formId", options);

`

- Set the required property in the FormValidator rules collection. Here, the min property of slider that sets the minimum value in the slider component is set, and it has hidden input as enable `validateHidden` property is set to true.

Copyright © 2001 -2024 Syncfusion Inc.

`ts

// Slider element

<ejs-slider id="default" name="slider"></ejs-slider>

// sets required property in the FormValidator rules collection

let options: FormValidatorModel = {

rules: {

'default': {

validateHidden: true,

min: [6, "You must select value greater than 5"]

}

}

};

`

Form validation is done either by ID or name value of the slider component. Above ID of the slider is used to validate it.

Using slider name: Render slider with name attribute. In the following code snippet, name attribute value of ?slider? is used for form validation.

`ts

// Slider element

<ejs-slider id="default" name="slider"></ejs-slider>

// sets required property in the FormValidator rules collection

let options: FormValidatorModel = {

rules: {

'slider': {

validateHidden: true,

min: [6, "You must select value greater than 5"]

}

}

};

`

- Validate the form using validate method, and it validates the slider value with the defined rules collection and returns the result. If user selects the value less than the minimum value, form will not submit.

`ts

Copyright © 2001 -2024 Syncfusion Inc.

formObj.validate();

`

- Slider validation can be done during value changes in slider. Refer to the following code snippet.

`ts

// change event handler for slider

onChanged(args) {

formObj.validate();

}

`

The FormValidator has following default validation rules, which are used to validate the Slider component.

| Rules | Description | Example |

| ---------- | ---------- | ---------- |

| max | Slider component must have value less than or equal to max number | if max: 3, **3** is valid and **4** is invalid |

| min | Slider component must have value greater than or equal to min number | if min: 4, **5** is valid and **2** is invalid |

| regex | Slider component must have valid value in regex format | if regex: '/4/, **4** is valid and **1** is invalid |

| range | Slider component must have value between range number | if range: [4,5], **4** is valid and **6** is invalid |

APP.VUE

```
<template>
   <div id='app'>
     <div class="col-lg-12 control-section">
            <div class="content-wrapper" style="margin-bottom:
25px;overflow-x: hidden">
                <div class="form-title">
                    <span>Min</span>
                </div>
                <form id="formMinId" class="form-horizontal">
                    <div class="form-group">
                        <div class="e-float-input">
                            <ejs-slider id="min-slider" name="min-slider"
type='MinRange'  value= 30 :ticks='ticks' v-on:changed='onMinChanged'></ejs-
slider>
                        </div>
                    </div>
                </form>
                <div class="form-title">
                    <span>Max</span>
                </div>
```

```
                <form id="formMaxId" class="form-horizontal">
                    <div class="form-group">
                        <div class="e-float-input">
                            <ejs-slider id="max-slider" name="max-slider"
type='MinRange'  value= 30 :ticks='ticks' v-on:changed='onMaxChanged'></ejs-
slider>
                        </div>
                    </div>
                </form>
                <div class="form-title">
                    <span>Value</span>
                </div>
                <form id="formValId" class="form-horizontal">
                    <div class="form-group">
                        <div class="e-float-input">
                            <ejs-slider id="val-slider" name="val-slider"
type='MinRange'  value= 30 :ticks='ticks' v-on:changed='onValChanged'></ejs-
slider>
                        </div>
                    </div>
                </form>
                <div class="form-title">
                    <span>Range</span>
                </div>
                <form id="formRangeId" class="form-horizontal">
                    <div class="form-group">
                        <div class="e-float-input">
                            <ejs-slider id="range-slider" name="range-
slider" type='MinRange'  value= 30 :ticks='ticks' v-
on:changed='onRangeChanged'></ejs-slider>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</template>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
    #app {
        height: 40px;
        position: absolute;
        width: 98%;
    }
    .highcontrast form {
        background: #000000
    }
    /* csslint ignore:start */
    .highcontrast label.e-custom-label,
    .highcontrast label.e-float-text,
    .highcontrast label.salary,
    .highcontrast input::placeholder,
    .highcontrast .e-float-input label.e-float-text {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        color: #fff !important;
        line-height: 2.3;
    }
    /* csslint ignore:end */
    .e-error,
    .e-float-text {
        font-weight: 500;
    }
    table,
    td,
    th {
        padding: 5px;
    }
    .form-horizontal .form-group {
        margin-left: 0;
        margin-right: 0;
    }
    form {
        border: 1px solid #ccc;
        box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.36);
        border-radius: 5px;
        background: #f9f9f9;
        padding: 23px;
        padding-bottom: 20px;
        margin: auto;
        max-width: 650px;
    }
    .form-title {
        width: 100%;
        text-align: center;
        padding: 10px;
        font-size: 16px;
        font-weight: 600;
        color: black;
    }
</style>
<script>
import Vue from 'vue';
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
export default {
  data: function() {
   return {
            ticks:{ placement: 'Before', largeStep: 20, smallStep: 5,
showSmallTicks: true }
    };
  },
  methods:{
        onMinChanged(args) {
            // sets required property in the FormValidator rules collection
            let minOptions = {
            rules: {
                'min-slider': {
                validateHidden: true,
                min: [40, "You must select value greater than or equal to
40"]
```

```
                }
            }
            };
            // Initialize Form validation
            let formMinObj;
            formMinObj = new FormValidator("#formMinId", minOptions);
            // validate the slider value in the form
            formMinObj.validate();
        },
        onMaxChanged(args) {
            // sets required property in the FormValidator rules collection
            let maxOptions = {
            rules: {
                'max-slider': {
                validateHidden: true,
                max: [40, "You must select value less than or equal to 40"]
                }
            }
            };
                // Initialize Form validation
            let formMaxObj;
            formMaxObj = new FormValidator("#formMaxId", maxOptions);
            // validate the slider value in the form
            formMaxObj.validate();
        },
        onValChanged(args) {
            // sets required property in the FormValidator rules collection
            let valOptions = {
            rules: {
                'val-slider': {
                validateHidden: true,
                regex: [/40/, "You must select value equal to 40"]
                }
            }
            };
                // Initialize Form validation
            let formValObj;
            formValObj = new FormValidator("#formValId", valOptions);
            // validate the slider value in the form
            formValObj.validate();
        },
        onRangeChanged(args) {
            // sets required property in the FormValidator rules collection
            let rangeOptions = {
            rules: {
                'range-slider': {
                validateHidden: true,
                range: [40, 80, "You must select values between 40 and 80"]
                }
            }
            };
                // Initialize Form validation
            let formRangeObj;
            formRangeObj = new FormValidator("#formRangeId", rangeOptions);
            // validate the slider value in the form
            formRangeObj.validate();
        }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/range-slider/thumb-cs2" %}

Show slider from hidden state in Vue Range slider component

This section demonstrates how-to render the Slider component in hidden state and make it visible in button click. We can initialize Slider in hidden state by setting the display as none.

In the sample, by clicking on the button, we can make the Slider visible from hidden state, and we must also call the refresh method of the Slider to render it properly based on its original dimensions.

APP.VUE

```
<template>
    <div id="app">
      <button type="button" id="btn" class="e-control e-btn" v-
on:click="onClick">Button</button>
      <div id="wrap">
        <ejs-slider
          id="slider"
          :min="minVal"
          :max="maxVal"
          :value="value"
          :tooltip="tooltip"
          :ticks="ticks"
          showButtons="true"
          :step="stepVal"
          v-on:renderingTicks="onRenderingTicks"
          v-on:tooltipChange="onTooltipChange"
        ></ejs-slider>
      </div>
    </div>
  </template>
  <style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
  #app {
    height: 40px;
    position: absolute;
    width: 98%;
  }
  #wrap {
    box-sizing: border-box;
    height: 260px;
    margin: 0 auto;
    padding: 30px 10px;
    width: 460px;
    display: none;
  }
  </style>
  <script>
  import Vue from "vue";
```

```
  import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
  Vue.use(SliderPlugin);
  export default {
    data: function() {
      return {
        tooltip: { placement: "Before", isVisible: true },
        minVal: new Date(2013, 6, 13, 11).getTime(),
        maxVal: new Date(2013, 6, 13, 17).getTime(),
        value: new Date(2013, 6, 13, 13).getTime(),
        stepVal: 3600000,
        // Slider ticks customization
        ticks: { placement: "After", largeStep: 2 * 3600000 }
      };
    },
    methods: {
      onTooltipChange(args) {
        let totalMiliSeconds = Number(args.text);
        let custom = { hour: "2-digit", minute: "2-digit" };
        args.text = new Date(totalMiliSeconds).toLocaleTimeString(
          "en-us",
          custom
        );
      },
      onRenderingTicks(args) {
        let totalMiliSeconds = Number(args.value);
        let custom = { hour: "2-digit", minute: "2-digit" };
        args.text = new Date(totalMiliSeconds).toLocaleTimeString(
          "en-us",
          custom
        );
      },
      onClick: function (event){
        let slider = document.getElementById("wrap");
        let ticks = document.getElementById("slider");
        slider.style.display = "block";
        ticks.ej2_instances[0].refresh();
      }
    }
  };
  </script>
```

{% previewsample "page.domainurl/code-snippet/range-slider/hidden-slider-cs1" %}

Reversible Range Slider in Vue

You can create a Range Slider rendered with values in reverse order by setting the min property to the maximum value and the max property to the minimum value. An example of how to achieve a reversible Range Slider is shown below

APP.VUE

```
<template>
    <div id="app">
    <ejs-slider id='reverse' :orientation="orientation" :type="type"
:min="min" :max="max" :tooltip="tooltip" :ticks="ticks" ></ejs-slider>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { SliderPlugin } from "@syncfusion/ej2-vue-inputs";
Vue.use(SliderPlugin);
export default {
  data() {
    return {
        ticks: { placement: 'Before', largeStep: 20, smallStep: 5,
showSmallTicks: true },
        tooltip: { placement: 'Before', isVisible: true, showOn: 'Always' },
        type: 'Range',
        // vertical orientation
        orientation: 'Vertical',
        // Set maximum value to min
        min: 100,
        // Set minimum value to max
        max: 0,
        // Slider current value
        value: [30, 70]
    };
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #app {
    color: #008cff;
    height: 340px;
    left: 30%;
    position: absolute;
    width: 50%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/range-slider/reversible-cs1" %}

Reversible order can be achieved with Horizontal orientation Range Slider by setting enableRtl as true.

Rating

Getting Started with the Vue Rating Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Rating component using the Composition API / Options API.

Dependencies

The list of dependencies required to use the Rating component in your application is given as follows:

`js

|-- @syncfusion/ej2-vue-inputs

|-- @syncfusion/ej2-vue-base

Copyright © 2001 -2024 Syncfusion Inc.

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-inputs

|-- @syncfusion/ej2-popups

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

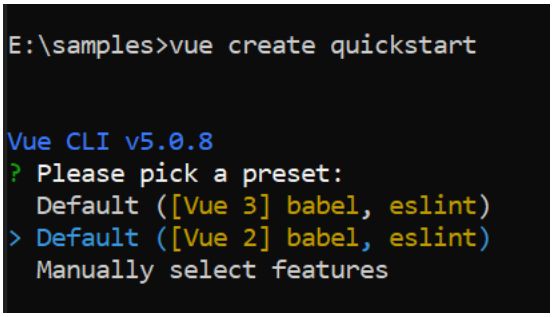
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option `Default ([Vue 2] babel, eslint)` from the menu.



Once the `quickstart` project is set up with default settings, proceed to add Syncfusion components to the project.

Install Syncfusion `Rating` packages using below command.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Rating component as an example. Install the @syncfusion/ej2-vue-inputs package by running the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Rating component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Rating component using Composition API or Options API:

1\. First, import and register the Rating component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { RatingComponent as EjsRating } from "@syncfusion/ej2-vue-inputs";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { RatingComponent } from "@syncfusion/ej2-vue-inputs";
export default {
components: {
'ejs-rating': RatingComponent
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

2\. In the template section define the Rating component.

~/SRC/APP.VUE

```
<template>
<div class='wrap'>
<ejs-rating id="rating"></ejs-rating>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating"></ejs-rating>
    </div>
</template>
<script setup>
import { RatingComponent as EjsRating } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating"></ejs-rating>
    </div>
</template>
<script>
import { RatingComponent } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
  components: {
    'ejs-rating': RatingComponent
  }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/rating/getting-started-cs1" %}

Value

You can set the rating value by using the value property.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3.0"></ejs-rating>
    </div>
</template>
<script setup>
import { RatingComponent as EjsRating } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3.0"></ejs-rating>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { RatingComponent } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
  components: {
    'ejs-rating': RatingComponent
  }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/value-cs1" %}

Getting Started with Syncfusion Rating Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Rating component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

Copyright © 2001 -2024 Syncfusion Inc.

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Rating component as an example. To use the Vue Rating component in the project, the @syncfusion/ej2-vue-inputs package needs to be installed using the following command:

Adding Syncfusion Rating package in the application

Syncfusion Vue packages are maintained in the npmjs.com registry. The Rating component will be used in this example. To install it use the following command.

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Adding CSS reference for Syncfusion Vue Rating component

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Button component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
</style>
```

Adding Syncfusion Vue Rating component in the application

Follow the below steps to add the Vue Rating component using Composition API or Options API:

1. First, import and register the Rating component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

Copyright © 2001 -2024 Syncfusion Inc.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { RatingComponent as EjsRating } from "@syncfusion/ej2-vue-inputs";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { RatingComponent } from "@syncfusion/ej2-vue-inputs";
export default {
name: "App",
components: {
"ejs-rating": RatingComponent
}
}
</script>
```

2. Add the component definition in template section.

~/SRC/APP.VUE

```
<template>
<ejs-rating id="rating"></ejs-rating>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-rating id="rating"></ejs-rating>
</template>
<script setup>
import { RatingComponent as EjsRating } from "@syncfusion/ej2-vue-inputs";
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-rating id="rating"></ejs-rating>
</template>
<script>
import { RatingComponent } from "@syncfusion/ej2-vue-inputs";
export default {
name: "App",
components: {
"ejs-rating": RatingComponent
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Precision modes in Vue Rating component

You can use the precision property of the rating component to provide ratings with varying levels of precision.

The precision types of Rating are as follows:

- Full: The rating is increased in whole number increments. For example, if the current rating is 2, the next possible ratings are 3, 4, and so on.
- Half: The rating is increased in increments of 0.5 (half). For example, if the current rating is 2.5, the next possible ratings are 3, 3.5, 4, and so on.
- Quarter: The rating is increased in increments of 0.25 (quarter). For example, if the current rating is 3.75, the next possible ratings are 4, 4.25, 4.5, and so on.
- Exact: The rating is increased in increments of 0.1. For example, if the current rating is 3.9, the next possible ratings are 4, 4.1, 4.2, and so on.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div class='wrap'>
        <label>Full Precision</label><br/>
        <ejs-rating id="rating1" value="3" precision="Full"></ejs-
rating><br/>
        <label>Half Precision</label><br/>
        <ejs-rating id="rating2" value="2.5" precision="Half"></ejs-
rating><br/>
        <label>Quarter Precision</label><br/>
        <ejs-rating id="rating3" value="3.75" precision="Quarter"></ejs-
rating><br/>
        <label>Exact Precision</label><br/>
        <ejs-rating id="rating4" value="2.3" precision="Exact"></ejs-
rating><br/>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/events-cs2" %}

Labels in Vue Rating component

You can use the showLabel property to display a label that shows the current value of the rating. When the showLabel property is set to true, a label will be displayed.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" :showLabel="true"></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/labels/showlabel-cs1" %}

Label position

The rating component allows you to place the label on the top, bottom, left, or right side of the rating using the labelPosition property.

The following label positions are supported:

- Top: The label is placed on the top of the rating.
- Bottom: The label is placed on the bottom of the rating.
- Left: The label is placed on the left side of the rating.
- Right: The label is placed on the right side of the rating.

APP.VUE

```
<template>
    <div class='wrap'>
        <label>Left Label Position</label><br/>
        <ejs-rating id="rating1" value="3" :showLabel="true"
labelPosition="Left"></ejs-rating><br/>
        <label>Right Label Position</label><br/>
        <ejs-rating id="rating2" value="3" :showLabel="true"
labelPosition="Right"></ejs-rating><br/>
        <label>Top Label Position</label><br/>
        <ejs-rating id="rating3" value="3" :showLabel="true"
labelPosition="Top"></ejs-rating><br/>
        <label>Bottom Label Position</label><br/>
        <ejs-rating id="rating4" value="3" :showLabel="true"
labelPosition="Bottom"></ejs-rating><br/>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/labels/label-positions-cs1" %}

Label template

You can use the labelTemplate tag directive to specify a custom template for the Label of the rating. The current value of the rating will be passed as the value property in the template context when building the content of the label. This allows you to include dynamic information about the rating in the template.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" :showLabel="true"
:labelTemplate="labelTemplate" ></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {
    data() {
        return {
            labelTemplate: function(e) {
                 return { template: Vue.component("labelTemplate", {
                    template: '<span>{{data.value}} out of 5</span>',
                    data() {
                        return {
                            data: {}
                        };
                    }
                })}
            }
        }
    };
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/labels/label-template-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Tool tip in Vue Rating component

The rating component supports tooltip to show additional information in rating items by setting the showTooltip property. If enabled, the tooltip appears when the user hovers over a rating item.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" :showTooltip="true" ></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/tooltip/showtooltip-cs1" %}

Tooltip template

You can use the tooltipTemplate tag directive to specify a custom template for the tooltip of the rating. The current value of the rating will be passed as the value property in the template context when building the content of the tooltip. This allows you to include dynamic information about the rating in the template.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" :showTooltip="true"
:tooltipTemplate="tooltipTemplate"></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
var ratingTooltipTemplate = Vue.component("tooltipTemplate", {
  template: `<b v-if='data.value==1'>Angry</b>
             <b v-else-if='data.value==2'>Sad</b>
             <b v-else-if='data.value==3'>Neutral</b>
             <b v-else-if='data.value==4'>Good</b>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <b v-else>Happy</b>`,
  data() {
    return {
        data: {}
    };
  }
});
export default {
    data() {
        return {
            tooltipTemplate: function(e) {
                return {
                    template: ratingTooltipTemplate
                }
            }
        }
    };
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/tooltip/tooltip-template-cs1" %}

Tooltip customization

You can customize the appearance of the tooltips using the cssClass property of the rating component and by defining the custom styles for tooltip elements like the below example.

You can find more information about customizing the appearance of the tooltip in the Tooltip Customization documentation.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" :showTooltip="true"
cssClass="customtooltip" ></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
/* To change the radius of the tooltip corners. */
.customtooltip.e-tooltip-wrap {
  border-radius: 3px;
}
/* To change the size of the tooltip content. */
.customtooltip.e-tooltip-wrap .e-tip-content {
  font-size:14px;
}
/* To change the border color and width for tooltip. */
.customtooltip.e-tooltip-wrap.e-popup {
  border: 2px solid #000000;
}
/* To change the color for arrow of the tooltip. */
.customtooltip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {
  border: 12px #9693
}
/* To change the top border color for arrow of the tooltip. */
.customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
  border-top: 9.5px solid #000000;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/tooltip/custom-tooltip-cs1" %}

Selection in Vue Rating component

The rating component allows users to rate something using a visual scale, and the selection state can be changed by the user clicking or tapping on the stars in the rating scale or through code. The rating component has a minimum value and a reset button, and provides customization options for the selected rating value and selection behavior.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" ></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/selection/default-cs1" %}

Min value

You can use the min property of the rating component to set the minimum possible rating value the user can select. If you set the min property to 2, then you will not be able to select a rating lower than 2.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" min="2" ></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/selection/min-cs1" %}

Single selection

You can use the enableSingleSelection property of the rating component to select only one item at a time. When the enableSingleSelection property is set to true, only the selected item will be considered to be in the selected state, while all other items will be unselected.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3"
:enableSingleSelection="true"></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/selection/single-selection-cs1" %}

Show or hide reset button

You can reset the rating value to its default by using the allowReset property of the rating component. When the allowReset property is set to true, a reset button will be shown that allows the user to reset the rating value to its default.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" :allowReset="true"></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/selection/allow-reset-cs1" %}

Templates in Vue Rating component

The rating component allows you to customize the appearance of the rating items using templates. You can use templates to specify a custom layout for the rating items, which can include any content you want. This allows you to create a more customized and interactive rating experience for the user.

The rating component supports below templates for item customization.

Copyright © 2001 -2024 Syncfusion Inc.

- emptyTemplate
- fullTemplate

Empty (unrated) symbol template

To customize the appearance of **unrated** items, you can use the emptyTemplate tag directive. It allows you to specify the desired custom content for the unrated items. The value and index are available in the template context for accessing information about the un-rated item.

If the fullTemplate is not defined, the emptyTemplate will be used as the default for both rated and unrated items. You can apply custom styles to differentiate between the rated and unrated states of the items.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" emptyTemplate="<span
class='custom-font sf-rating-heart'></span>" ></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
.e-rating-container .custom-font {
  /* To add the background color for the font icon. */
  background: linear-gradient(to right, rgb(254,87,133,255) var(--rating-
value), transparent var(--rating-value));
  /* To clip the background to the icon (text) alone. */
  background-clip: text;
  -webkit-background-clip: text;
  /* To make the background color visible instead of font color. */
  -webkit-text-fill-color: transparent;
  /* To provide a border for font icon. */
  -webkit-text-stroke: 1px rgb(254,87,133,255);
}
@font-face {
  font-family: "rating";
  src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1uSfQAAAEoAAAAVmNtYXDnEudaAAABjAAAADhnbHlm4Li
FsgAAAcwAAAJsaGVhZCKCSVkAAADQAAAANmhoZWEIUQQEAAAArAAAACRobXR4DAAAAAAAAYAAAAA
MbG9jYQCMATYAAAHEAAAACG1heHABDwCZAAABCAAAACBuYW1l75Kp8wAABDgAAAIZcG9zdDjyU90
AAAZUAAAANwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAwABAAAAAQAA2T6Kh18
PPPUACwQAAAAAN+4AkEAAAAA37gCQQAAAAAD9APaAAAACAACAAAAAAAAAAEAAAADAI0AAgAAAAA
```

```
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wHnAgQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABAAkAAAABAAEAAEAAOcC//8AAOcB//8AAAABAAQAAAA
CAAEAAAAAAIwBNgABAAAAAAAPzA9oAfAAAEw8WFR8PPw41Lx4jDwwvDw8GqAwMDAsKCgoJCAgIBwY
GBQUEBAMCAgEBAQECAwMEBQULFSMhOVJliOxTOSEdFg0IBQQDAwIBAQEBAgIDBAQFBQYGBwgICAk
KCgoLDAwMDAwMDQwMDQwZGBgYFxUVFBIRCAgGBwkLCwwNDg4PEBAQEREREhEODg4ODg4NA8IGBwc
ICAkJCgoKCwsMCwwNDA0MDQ0ODQ0ODQ0ODQ0ODQ0NDRUiMCtEX26P/V5FKycjFhQNDQ0ODQ0ODQ0NDgw
NDQwNCwwMCwoLCgoJCAkIBwcGBQUEAwMCAQECBQYJCw4PERMKCgsMEQ8PDQ0LCwoICAYFBAMCAQE
BAgIEBAUAAgAAAAD9APFAAMAjAAANzMRIwEPAxUXDwwRMzcfBDcXPwo9AS8FPwsvCDc1Pwg1LwU
1Pw01LwkHJT8ENS8LIw8BDK2tAfkCCgQBAQEBGCERERITIgkJKBAGIQc1Bx45k9sOBQgLDQsJBQM
EAgIECQYCAQEBAw4ECQgGBwMDAQEBAQMDAwkCAQEDFgsFBAQDAwICAgQECgEBAQQKBwcGBQUEAwM
BAQEBBAUHCQUFBQYR/q0PCQQDAgEBAwMKDBUDBwYMCw0HB1oBhwHeAQUDA3YfCgQsOh0bHBovCQg
bDP6KAQEfAwEBAQIBAQMGCgoMBggICAUICQgLBQQEBAUDBgMHCAgMCAcIBwYGBgUFCQQCBgIEDAk
GBQYHCQkKCQgIBwsEAgUDAgQEBAUFBgcHCAcGBgYGCgkIBgICAQEBAUYxGRobDQ0MDQsiHjEEBAI
EAQECAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAEABgABAAEAAAAAAAIABwAHAAEAAAAAAAMABgA
OAAEAAAAAAAQABgAUAAEAAAAAAAUACwAaAAEAAAAAAAYABgAlAAEAAAAAAAoALAArAAEAAAAAAAs
AEgBXAAMAAQQJAAAAAgBpAAMAAQQJAAEADABrAAMAAQQJAAIADgB3AAMAAQQJAAMADACFAAMAAQQ
JAAQADACRAAMAAQQJAAUAFgCdAAMAAQQJAAYADACzAAMAAQQJAAoAWAC/AAMAAQQJAAsAJAEXIHJ
hdGluZ1JlZ3VsYXJyYXRpbmdyYXRpbmdWZXJzaW9uIDEuMHJhdGluZ0ZvbnQgZ2VuZXJhdGVkIHV
zaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAcgBhAHQAaQB
uAGcAUgBlAGcAdQBsAGEAcgByAGEAdABpAG4AZwByAGEAdABpAG4AZwBWAGUAcgBzAGkAbwBuACA
AMQAuADAAcgBhAHQAaQBuAGcARgBvAG4AdAAgAGcAZQBuAGUAcgBhAHQAZQBkACAAdQBzAGkAbgB
nACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4
AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMAbwBtAAAAAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAwECAQMBBAAFaGVhcnQFdGhlbWIAAAA=) format('truetype');
  font-weight: normal;
  font-style: normal;
}
[class^="sf-rating-"], [class*=" sf-rating-"] {
  font-family: "rating" !important;
  speak: none;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.sf-rating-heart:before {
  content: "\e702";
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/templates/empty-template-cs1" %}

The current value of the rating item available in the template context as `value` and in the rating item element as CSS Variable(`--rating-value`) can be used to support precision in templates.

Full (rated) symbol template

To customize the appearance of **rated** items in the rating component, you can use the `fullTemplate` tag directive. This directive allows you to specify a custom layout for the rated items, which can include any content you desire. The `value` and `index` are available in the template context for accessing information about the rated item.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" emptyTemplate="<span
class='custom-font sf-icon-empty-star'></span>" fullTemplate="<span
class='custom-font sf-icon-fill-star'></span>" ></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
.e-rating-container .custom-font {
  /* To change the icon font color. */
  color: rgb(255, 215, 0);
}
@font-face {
  font-family: 'rating-template';
  src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tSfMAAAEoAAAAVmNtYXDnEOdaAAABjAAAADhnbHlm+ic
DjQAAAcwAAAE0aGVhZCK49ucAAADQAAAANmhoZWEIUQQEAAAArAAAACRobXR4DAAAAAAAAYAAAAA
MbG9jYQAcAJoAAAHEAAAACG1heHABDwBkAAABCAAAACBuYW1lmYExxgAAAwAAAAKFcG9zdCH169Q
AAAWIAAAAQAABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAwABAAAAAQAAgPX4jF8
PPPUACwQAAAAAAN/TWPsAAAAA39NY+wAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAADAFgAAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAQQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABAAkAAAABAAEAAEAAOcB//8AAOcA//8AAAABAAQAAAA
BAAIAAAAAABwAmgABAAAAAAP0A/QACQAAAQUTAyUFAxMlAwFn/qX6OwE1ATU7+v6lmQKsNf7//pa
srAFqAQE1AUgAAAIAAAAAA/QD5AAdAFcAAAEfBAUPAxUTLwEjDwETNS8DJT8EJwMFDwQVHwIDBx8
EMzclBRczPwU1Az8CNS8DJQMvBisBDwUCYAICBgMHASCwBAMCGuoHCAjpGgIDBLEBHgcGBgJiHXb
+uQgHBgQBAgTUHgECBAUHCAkIAQ4BDgcJBAQEBwQDHdMEAgMFBgf+tHYDAgMEBAQEBQUEBAQEAwI
CnwMDBgIDNbAGBgYE/uCBAgKBAR0HBgYGsDQCBAYD3Fr+9jwDBAcHBQgIB9T+twUIBwcEAwKVlQI
BAgIFBwgJAUnUBwgJCAcFBD0BCgQDBAICAgEBAgICBAMAAAAAEgDeAAEAAAAAAAAAQAAAAEAAAA
AAAEADwABAAEAAAAAAAIABwAQAAEAAAAAAAMADwAXAAEAAAAAAAQADwAmAAEAAAAAAAUACwA1AAE
AAAAAAAYADwBAAAEAAAAAAAoALABPAAEAAAAAAAsAEgB7AAMAAQQJAAAAAgCNAAMAAQQJAAEAHgC
PAAMAAQQJAAIADgCtAAMAAQQJAAMAHgC7AAMAAQQJAAQAHgDZAAMAAQQJAAUAFgD3AAMAAQQJAAY
AHgENAAMAAQQJAAoAWAErAAMAAQQJAAsAJAGDIHJhdGluZy10ZW1wbGF0ZVJlZ3VsYXJyYXRpbmc
tdGVtcGxhdGVyYXRpbmctdGVtcGxhdGVWZXJzaW9uIDEuMHJhdGluZy10ZW1wbGF0ZUZvbnQgZ2V
uZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACA
AcgBhAHQAaQBuAGcALQB0AGUAbQBwAGwAYQB0AGUAUgBlAGcAdQBsAGEAcgByAGEAdABpAG4AZwA
tAHQAZQBtAHAAbABhAHQAZQByAGEAdABpAG4AZwAtAHQAZQBtAHAAbABhAHQAZQBWAGUAcgBzAGk
AbwBuACAAMQAuADAAcgBhAHQAaQBuAGcALQB0AGUAbQBwAGwAYQB0AGUARgBvAG4AdAAgAGcAZQB
uAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHI
AbwAgAFMAdAB1AGQaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMAbwBtAAAAAAI
```

```
AAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwECAQMBBAAJZmlsbC1zdGFyCmVtcHR5LXN
0YXIAAA==) format('truetype');
  font-weight: normal;
  font-style: normal;
}
[class^="sf-icon-"],
[class*=" sf-icon-"] {
  font-family: 'rating-template' !important;
  speak: none;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.sf-icon-fill-star:before {
  content: "\e700";
}
.sf-icon-empty-star:before {
  content: "\e701";
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/templates/full-template-cs1" %}

Using Emoji icon as rating symbol

You can use emojis of your choice as rating symbol by specifying them as template content within the `emptyTemplate` tag directive.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="4" :emptyTemplate="emptyTemplate"
:enableSingleSelection="true" :enableAnimation="false" ></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
var emojiTemplate = Vue.component("emptyTemplate", {
  template: `<span v-if='data.index==0' class='angry emoji'>😠</span>
             <span v-else-if='data.index==1' class='disagree
emoji'>☹</span>
             <span v-else-if='data.index==2' class='neutral
emoji'>😐</span>
             <span v-else-if='data.index==3' class='agree emoji'>🙂</span>
             <span v-else class='happy emoji'>😀</span>`,
  data() {
    return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        data: {}
    };
  }
});
export default {
    data() {
        return {
            emptyTemplate: function(e) {
                return {
                    template: emojiTemplate
                }
            }
        }
    };
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
/* To change the color of an unselected rating item. */
.e-rating-item-container:not(.e-rating-selected) .emoji {
  filter: grayscale(1);
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/templates/emoji-template-cs1" %}

Using SVG icon as rating symbol

You can use SVG icons of your choice as rating symbol by specifying them as template content within the `emptyTemplate` and `fullTemplate` tag directive

APP.VUE

```
<template>
  <div class='wrap'>
      <ejs-rating id="rating" value="4" :emptyTemplate="emptyTemplate"
:fullTemplate="fullTemplate" :enableAnimation="false" ></ejs-rating>
  </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
var emptyTemplate = Vue.component("emptyTemplate", {
template: `<svg width='35' height='25' className='e-rating-svg-icon'>
            <rect width='35' height='25' fill='transparent' strokeWidth='2'
stroke='rgb(173,181,189)'/>
          </svg>`,
data() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  return {
      data: {}
  };
}
});
var fullTemplate = Vue.component("fullTemplate", {
template: `<svg width="35" height="25" class="e-rating-svg-icon">
            <defs>
              <linearGradient :id="['grad'+data.index]" x1="0%" y1="0%"
x2="100%" y2="0%">
                <stop class="start" offset="0%" />
                <stop class="end" offset="100%" />
              </linearGradient>
            </defs>
            <rect width="35" height="25"
:fill="['url(#grad'+data.index+')']" style="stroke-
width:2;stroke:rgb(173,181,189)"/>
          </svg>`,
data() {
  return {
      data: {}
  };
}
});
export default {
  data() {
    return {
      emptyTemplate: function(e) {
        return {
            template: emptyTemplate
        }
      },
      fullTemplate: function(e) {
        return {
            template: fullTemplate
        }
      }
    }
  }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
margin: 50px auto;
text-align: center;
}
/* To change the size between items */
.e-rating-container .e-rating-item-container {
padding: 0px;
}
/* To set the gradient color */
.e-rating-svg-icon #grad0 .start {
stop-color: #FF0000;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.e-rating-svg-icon #grad0 .end,
.e-rating-svg-icon #grad1 .start {
stop-color: #ff5101;
}
.e-rating-svg-icon #grad1 .end,
.e-rating-svg-icon #grad2 .start {
stop-color: #ffc801;
}
.e-rating-svg-icon #grad2 .end,
.e-rating-svg-icon #grad3 .start {
stop-color: #dbe300;
}
.e-rating-svg-icon #grad3 .end,
.e-rating-svg-icon #grad4 .start {
stop-color: #8bc301;
}
.e-rating-svg-icon #grad4 .end {
stop-color: #4eaa01;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/templates/svg-template-cs1" %}

Using PNG image as rating symbol

You can use PNG images of your choice as rating symbol by specifying them as template content within the emptyTemplate and fullTemplate tag directives.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="4" emptyTemplate="<img src='star-
emptytemplate.png' widht='25' height='25' />" fullTemplate="<img src='star-
fulltemplate.png' widht='25' height='25' />" ></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/templates/png-template-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Events in Vue Rating component

This section describes the rating events that will be triggered when appropriate actions are performed. The following events are available in the rating component.

beforeItemRender

The rating component triggers the beforeItemRender event before rendering each rating item. The RatingItemEventArgs passed as an event argument provides the details of the item to be rendered.

```
<template>
<div class='wrap'>
<ejs-rating id="rating" :beforeItemRender="beforeItemRender"></ejs-rating>
</div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {
methods : {
beforeItemRender: function(args: RatingItemEventArgs) {
//Your required action here
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
margin: 50px auto;
text-align: center;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
`
```

created

The rating component triggers the created event when the rendering of the rating component is completed.

```
`
<template>
<div class='wrap'>
<ejs-rating id="rating" :created="created"></ejs-rating>
</div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {
methods : {
created: function() {
//Your required action here
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
margin: 50px auto;
text-align: center;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

`

onItemHover

The rating component triggers the onItemHover event when the rating item is hovered. The RatingHoverEventArgs passed as an event argument provides the details of the hovered item.

`

```
<template>
<div class='wrap'>
<ejs-rating id="rating" :onItemHover="onItemHover"></ejs-rating>
</div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {
methods : {
onItemHover: function(args: RatingHoverEventArgs) {
//Your required action here
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
margin: 50px auto;
text-align: center;
}
</style>
```

`

Copyright © 2001 -2024 Syncfusion Inc.

valueChanged

The rating component triggers the valueChanged event when the value of the rating is changed. The RatingChangedEventArgs passed as an event argument provides the details when value is changed.

```
<template>
<div class='wrap'>
<ejs-rating id="rating" :valueChanged="valueChanged"></ejs-rating>
</div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {
methods : {
valueChanged: function(args: RatingChangedEventArgs) {
//Your required action here
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
margin: 50px auto;
text-align: center;
}
</style>
```

Below example demonstrates the valueChanged event of the Rating component.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" :valueChanged="valueChanged"></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin, RatingChangedEventArgs  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {
    methods : {
        valueChanged: function(args: RatingChangedEventArgs) {
          alert("Previous Value:"+args.previousValue+"\nValue:"+args.value);
        }
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/events-cs1" %}

Appearance in Vue Rating component

You can also customize the appearance of rating component.

Items count

You can specify the number of rating items using the itemsCount property.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" itemsCount="8" value="3"></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/appearance/items-count-cs1" %}

Disabled

You can disable the rating component by using the disabled property. When the disabled property is set to true, the rating component will be disabled and the user will not be able to interact with it and a disabled rating component may have a different visual appearance than an enabled one.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" :disabled="true"></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/appearance/disabled-cs1" %}

Visible

You can use the visible property of the rating component to component the visibility of the component. When the visible property is set to true, the rating component will be visible on the page. When it is set to false, the component will be hidden.

APP.VUE

```
<template>
    <div class='wrap'>
        <button id='btn' @click="visible">Visible</button>
        <ejs-rating id="rating" value="3" :visible="true"
ref="rating"></ejs-rating>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { RatingPlugin  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {
    methods : {
        visible: function() {
            this.$refs.rating.visible = !this.$refs.rating.visible;
        }
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/appearance/visible-cs1" %}

Read only

You can use the readOnly property of the rating component to make the component non-interactive and prevent the user from changing the rating value.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" :readOnly="true"></ejs-rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/rating/appearance/read-only-cs1" %}

CssClass

You can customize the appearance of the rating component, such as by changing its colors, fonts, sizes, or other visual aspects by using the cssClass property.

Changing rating symbol border color

You can change the rating icon border color in rating component, you can use the cssClass property and set the text-stroke CSS property of .e-rating-icon to your desired border color.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" cssClass="custom-border" ></ejs-
rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
.e-rating-container.custom-border .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-border .e-rating-item-container .e-rating-icon {
  /* To change rating symbol border color */
  -webkit-text-stroke: 2px #ae9e9d;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/appearance/border-cs1" %}

Changing rated/un-rated symbol fill color

You can customize the fill colors of rated and un-rated icons in the rating component using the cssClass property and the linear-gradient color-stops in the background CSS property of .e-rating-icon. The **first** color-stop defines the rated fill color and the **second** defines the un-rated fill color.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" cssClass="custom-fill" ></ejs-
rating>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { RatingPlugin  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
.e-rating-container.custom-fill .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-fill .e-rating-item-container .e-rating-icon {
  /* To change rated symbol fill color and un-rated symbol fill color */
  background: linear-gradient(to right, #ffe814 var(--rating-value), #d8d7d4
var(--rating-value));
  background-clip: text;
  -webkit-background-clip: text;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/appearance/fill-color-cs1" %}

This will customize the rated fill color to #ffe814 and un-rated fill color to #d8d7d4. --rating-value in the linear-gradient provides the current value of the rating item.

Changing the item spacing

You can change the space between the rating items in rating component, by using the cssClass property and setting the margin / padding CSS property of .e-rating-item-container to your desired size.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" cssClass="custom-font" ></ejs-
rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
```

```
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
.e-rating-container.custom-font .e-rating-item-container {
  /* To change the size between items */
  margin: 0px 7px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/appearance/spacing-cs1" %}

Changing icon using CssClass

You can change the rating item icon in rating component, you can use the `cssClass` property and set the `content` CSS property of `.e-icons.e-star-filled:before` to your desired font icon.

APP.VUE

```
<template>
    <div class='wrap'>
        <ejs-rating id="rating" value="3" cssClass="custom-icon" ></ejs-
rating>
    </div>
</template>
<script>
import Vue from 'vue';
import { RatingPlugin  } from "@syncfusion/ej2-vue-inputs";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(RatingPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
.wrap {
  margin: 50px auto;
  text-align: center;
}
@font-face {
  font-family: 'custom-icon';
  src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSfQAAAEoAAAAVmNtYXDnEudXAAABiAAAADZnbHlmVIZ
rowAAAcgAAAEYaGVhZCK6KOUAAADQAAAANmhoZWEIUAQDAAAArAAAACRobXR4CAAAAAAAAYAAAAA
IbG9jYQCMAAAAAAHAAAAABm1heHABDQCJAAABCAAAACBuYW1lv3dY+QAAAuAAAAJVcG9zdN12Ynk
AAAU4AAAALwABAAAEAAAAAFwEAAAAAAAD8wABAAAAAAAAAAAAAAAAAAAAAgABAAAAAQAAEGWKhV8
PPPUACwQAAAAAAN/UcgcAAAAA39RyBwAAAAD8wPaAAAACAACAAAAAAAAAAEAAAACAH0AAQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnAgQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wL//wAA5wL//wAAAAEABAAAAAAEAAAA
AAAAAjAAAAAEAAAAAA/MD2gB8AAATDxYVHw8/DjUvHiMPDC8PDwaoDAwMCwoKCgkICAgHBgYFBQQ
EAwICAQEBAQIDAwQFBQsVIyE5UmWI7FM5IR0WDQgFBAMDAgEBAQECAgMEBAUFBgYHCAgICQoKCgs
MDAwMDAwNDAwNDBkYGBgXFRUUEhEICAYHCQsLDA0ODg8QEBARERESEQ4ODg4ODg0DwgYHBwgICQk
KCgoLCwwLDA0MDQwNDQ4NDQ4NDQ4NDQ0NFSIwK0Rfbo/9XkUrJyMWFA0NDQ4NDQ4NDQ0ODA0NDA0
```

```
LDAwLCgsKCgkICQgHBwYFBQQDAwIBAQIFBgkLDg8REwoKCwwRDw8NDQsLCggIBgUEAwIBAQECAgQ
EBQAAABIA3gABAAAAAAAAAAEAAAABAAAAAAABAAsAAQABAAAAAAACAAcADAABAAAAAAADAAsAEwA
BAAAAAAAEAAsAHgABAAAAAAAFAAsAKQABAAAAAAAGAAsANAABAAAAAAAKACwAPwABAAAAAAALABI
AawADAAEECQAAAAIAfQADAAEECQABABYAfwADAAEECQACAA4AlQADAAEECQADABYAowADAAEECQA
EABYAuQADAAEECQAFABYAzwADAAEECQAGABYA5QADAAEECQAKAFgA+wADAAEECQALACQBUyBjdXN
0b20taWNvblJlZ3VsYXJjdXN0b20taWNvbmN1c3RvbS1pY29uVmVyc2lvbiAxLjBjdXN0b20taWN
vbkZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXN
pb24uY29tACAAYwB1AHMAdABvAG0ALQBpAGMAbwBuAFIAZQBnAHUAbABhAHIAYwB1AHMAdABvAG0
ALQBpAGMAbwBuAGMAdQBzAHQAbwBtAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYwB
1AHMAdABvAG0ALQBpAGMAbwBuAEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4
AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwA
uAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAIBAgEDAAVoZWFydAAAAA==) format('truetype');
  font-weight: normal;
  font-style: normal;
}
.custom-icon .e-icons.e-star-filled {
  font-family: 'custom-icon' !important;
  speak: none;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.custom-icon .e-icons.e-star-filled:before {
  content: "\e702";
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rating/appearance/icon-cs1" %}

Accessibility in Vue Rating component

Accessibility is achieved in the rating component through WAI-ARIA standard and keyboard navigations. The rating component can be effectively accessed through assistive technologies such as screen readers.

Keyboard interaction

The rating component is interactive with below keyboard shortcuts.

| Keyboard shortcuts | Actions |

|------------|-------------------|

| Space | If a **Reset Button** is focused, resets the value to min value. |

| Arrow Up | Increases the value. |

| Arrow Left | Decreases the value and in RTL mode, increases the value. |

| Arrow Down | Decreases the value. |

| Arrow Right | Increases the value and in RTL mode, decreases the value. |

ARIA attribute

The following ARIA attributes are used in rating component based on its state.

Copyright © 2001 -2024 Syncfusion Inc.

| Properties | Functionality |

| ---------- | ---------- |

| role | This attribute is added to the div element to describe the actual role. |

| aria-label | Attribute provides the text label with some default description for the Rating and its items. |

| aria-valuemin | It defines the minimum value of rating. |

| aria-valuemax | It defines the maximum value of rating. |

| aria-valuenow | It defines the current value of rating. |

Ribbon

Getting Started with the Vue Ribbon Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Ribbon component

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies are required to use the Ribbon component in your application.

`js

|-- @syncfusion/ej2-vue-ribbon

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-data

|-- @syncfusion/ej2-buttons

|-- @syncfusion/ej2-popups

|-- @syncfusion/ej2-splitbuttons

|-- @syncfusion/ej2-inputs

|-- @syncfusion/ej2-lists

|-- @syncfusion/ej2-dropdowns

|-- @syncfusion/ej2-navigations

|-- @syncfusion/ej2-ribbon

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

Copyright © 2001 -2024 Syncfusion Inc.

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

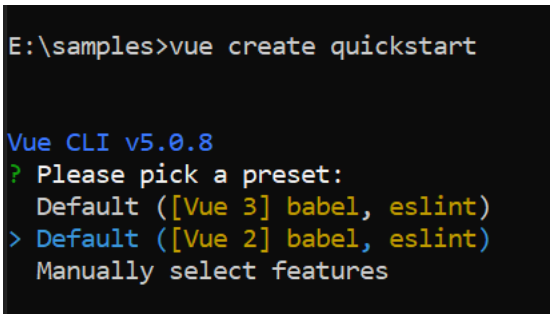
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Ribbon component as an example. Install the @syncfusion/ej2-vue-ribbon package by running the following command:

`bash

npm install @syncfusion/ej2-vue-ribbon --save

`

or

`bash

yarn add @syncfusion/ej2-vue-ribbon

`

Copyright © 2001 -2024 Syncfusion Inc.

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Ribbon component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Ribbon component using:

1\. First, import and register the Ribbon component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
export default {
components: {
'ejs-ribbon': RibbonComponent
}
}
</script>
```

2\. In the template section, define the Ribbon component.

~/SRC/APP.VUE

```
<template>
<ejs-ribbon id="ribbon"></ejs-ribbon>
</template>
```

Adding Ribbon Tab

In Ribbon, the options are arranged in tabs for easy access. You can use the e-ribbon-tab directive to define the ribbon tab like below.

~/SRC/APP.VUE

```
<template>
<ejs-ribbon id="ribbon">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-tabs>
<e-ribbon-tab header="Home"></e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import { RibbonComponent, RibbonTabDirective, RibbonTabsDirective } from
"@syncfusion/ej2-vue-ribbon";
export default {
components: {
'ejs-ribbon': RibbonComponent,
'e-ribbon-tab': RibbonTabDirective,
'e-ribbon-tabs': RibbonTabsDirective
}
};
</script>
```

Adding Ribbon Group

To define a ribbon group under each tab, you can use the e-ribbon-group directive like below. The orientation property of ribbon group defines whether the collection of items will be rendered column-wise or row-wise.

~/SRC/APP.VUE

```
<template>
<ejs-ribbon id="ribbon">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard" orientation="Row"></e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import { RibbonComponent, RibbonTabDirective, RibbonTabsDirective,
RibbonGroupDirective, RibbonGroupsDirective } from "@syncfusion/ej2-vue-
ribbon";
export default {
components: {
'ejs-ribbon': RibbonComponent,
'e-ribbon-tab': RibbonTabDirective,
'e-ribbon-tabs': RibbonTabsDirective,
'e-ribbon-groups': RibbonGroupsDirective,
'e-ribbon-group': RibbonGroupDirective
}
};
</script>
```

Adding Ribbon Items

You can use the e-ribbon-collection directive to define each ribbon collection that contains one or more items. To define each ribbon item, you can use the e-ribbon-item directive and the type property

Copyright © 2001 -2024 Syncfusion Inc.

to specify the type of component to be rendered, like a button, a drop-down button, a combo box, and more.

~/SRC/APP.VUE

```
<template>
<ejs-ribbon id="ribbon">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard" orientation="Column">
<e-ribbon-collections>
<e-ribbon-collection id="paste-collection">
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :splitButtonSettings="pasteSettigs"></e-
ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
<e-ribbon-collection id="cutcopy-collection">
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton"></e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="copyButton"></e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import { RibbonComponent, RibbonTabDirective, RibbonTabsDirective,
RibbonGroupDirective, RibbonGroupsDirective, RibbonCollectionDirective,
RibbonCollectionsDirective, RibbonItemDirective, RibbonItemsDirective } from
"@syncfusion/ej2-vue-ribbon";
export default {
components: {
'ejs-ribbon': RibbonComponent,
'e-ribbon-tab': RibbonTabDirective,
'e-ribbon-tabs': RibbonTabsDirective,
'e-ribbon-group': RibbonGroupDirective,
'e-ribbon-groups': RibbonGroupsDirective,
'e-ribbon-collection': RibbonCollectionDirective,
'e-ribbon-collections': RibbonCollectionsDirective,
'e-ribbon-item': RibbonItemDirective,
'e-ribbon-items': RibbonItemsDirective
},
data: function () {
return {
pasteSettigs:{
iconCss: "e-icons e-paste", content: "Paste",
items: [{ text: "Keep Source Format" }, { text: "Merge format" }, { text:
"Keep text only" }]
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
}
};
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

~/SRC/APP.VUE

```
<template>
  <ejs-ribbon :fileMenu="fileSettings">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Clipboard" groupIconCss="e-icons e-paste"
:showLauncherIcon="true" >
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="SplitButton"
:allowedSizes="largeSize" :splitButtonSettings="pasteSettigs" >
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button" :buttonSettings="copyButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button"
:buttonSettings="formatButton">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
          <e-ribbon-group header="Font" orientation="Row"
:enableGroupOverflow="true" :isCollapsible="false" groupIconCss="e-icons e-
bold" cssClass="font-group" >
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="ComboBox"
:comboBoxSettings="styleOptions" >
                  </e-ribbon-item>
                  <e-ribbon-item type="ComboBox"
:comboBoxSettings="sizeOptions">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="ColorPicker"
displayOptions="Simplified" :allowedSizes="smallSize"
:colorPickerSettings="colorPicker" >
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    </e-ribbon-item>
                    <e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="boldButton" >
                    </e-ribbon-item>
                    <e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="italicButton">
                    </e-ribbon-item>
                    <e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="underlineButton" >
                    </e-ribbon-item>
                    <e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="strikethroughButton">
                    </e-ribbon-item>
                    <e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="caseButton" >
                    </e-ribbon-item>
                  </e-ribbon-items>
                </e-ribbon-collection>
              </e-ribbon-collections>
            </e-ribbon-group>
            <e-ribbon-group header="Editor" :isCollapsible="false"
groupIconCss="e-icons e-edit" >
              <e-ribbon-collections>
                <e-ribbon-collection>
                  <e-ribbon-items>
                    <e-ribbon-item type="Button" :allowedSizes="largeSize"
:buttonSettings="editButton">
                    </e-ribbon-item>
                  </e-ribbon-items>
                </e-ribbon-collection>
              </e-ribbon-collections>
            </e-ribbon-group>
          </e-ribbon-groups>
        </e-ribbon-tab>
        <e-ribbon-tab header="Insert">
          <e-ribbon-groups>
            <e-ribbon-group header="Tables" :isCollapsible=false>
              <e-ribbon-collections>
                <e-ribbon-collection>
                  <e-ribbon-items>
                    <e-ribbon-item type="DropDown" :allowedSizes="largeSize"
:dropDownSettings="tableSettings">
                    </e-ribbon-item>
                  </e-ribbon-items>
                </e-ribbon-collection>
              </e-ribbon-collections>
            </e-ribbon-group>
            <e-ribbon-group header="Illustrations" id="illustration"
orientation="Row" :enableGroupOverflow=true groupIconCss="e-icons e-image">
              <e-ribbon-collections>
                <e-ribbon-collection>
                  <e-ribbon-items>
                    <e-ribbon-item type="Button"
:buttonSettings="chartSettings">
                    </e-ribbon-item>
                  </e-ribbon-items>
                </e-ribbon-collection>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            </e-ribbon-collections>
          </e-ribbon-group>
          <e-ribbon-group header="Media" :isCollapsible=false>
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Template"
:itemTemplate="'ribbonTemplate'">
                    <template v-slot:ribbonTemplate = "{data}">
                      <span v-bind:class="'ribbonTemplate ' +
data.activeSize"><span class="e-icons e-video"></span><span
class="text">Video</span></span>
                    </template>
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
      <e-ribbon-tab header="View">
        <e-ribbon-groups>
          <e-ribbon-group header="Views" orientation="Row" groupIconCss="e-
icons e-print">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button"
:buttonSettings="printSettings">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button"
:buttonSettings="layoutSettings">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
          <e-ribbon-group header="show" :isCollapsible=false>
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="CheckBox"
:checkBoxSettings="rulerSettings">
                  </e-ribbon-item>
                  <e-ribbon-item type="CheckBox"
:checkBoxSettings="gridSettings">
                  </e-ribbon-item>
                  <e-ribbon-item type="CheckBox"
:checkBoxSettings="navigationSettings">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  </ejs-ribbon>
</template>
<script>
  import { RibbonFileMenu, RibbonItemSize, RibbonComponent,
RibbonTabDirective, RibbonTabsDirective, RibbonGroupDirective,
RibbonGroupsDirective, RibbonCollectionDirective,
RibbonCollectionsDirective, RibbonItemDirective, RibbonItemsDirective,
RibbonColorPicker } from "@syncfusion/ej2-vue-ribbon";

  export default {
    components: {
      'ejs-ribbon': RibbonComponent,
      'e-ribbon-tab': RibbonTabDirective,
      'e-ribbon-tabs': RibbonTabsDirective,
      'e-ribbon-group': RibbonGroupDirective,
      'e-ribbon-groups': RibbonGroupsDirective,
      'e-ribbon-collection': RibbonCollectionDirective,
      'e-ribbon-collections': RibbonCollectionsDirective,
      'e-ribbon-item': RibbonItemDirective,
      'e-ribbon-items': RibbonItemsDirective
    },
    provide: {
      ribbon: [RibbonFileMenu, RibbonColorPicker]
    },
    data: function () {
      return {
        largeSize: RibbonItemSize.Large,
        smallSize: RibbonItemSize.Small,
        fileSettings: {
          visible: true,
          menuItems: [
            { text: "New", iconCss: "e-icons e-file-new", id: "new" },
            { text: "Open", iconCss: "e-icons e-folder-open", id: "open" },
            { text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
            { text: "Save as", iconCss: "e-icons e-save", id: "save" }
          ]
        },
        pasteSettigs:{
          iconCss: "e-icons e-paste", content: "Paste",
          items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
        },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter" },
        styleOptions: {
          dataSource: ["Algerian", "Arial", "Calibri", "Cambria", "Cambria
Math", "Courier New", "Candara", "Georgia", "Impact", "Segoe Print", "Segoe
Script", "Segoe UI", "Symbol", "Times New Roman", "Verdana", "Windings" ],
          index: 3,
          width: "150px",
          allowFiltering: true
        },
        sizeOptions: {
          dataSource: ["8", "9", "10", "11", "12", "14", "16", "18", "20",
"22", "24", "26", "28", "36", "48", "72", "96" ],
```

```
          index: 4,
          width: "65px"
        },
        boldButton:  { iconCss: "e-icons e-bold", content: "Bold", isToggle:
"true" },
        italicButton:  { iconCss: "e-icons e-italic", content: "Italic",
isToggle: "true" },
        underlineButton:  { iconCss: "e-icons e-underline", content:
"Underline", isToggle: "true" },
        strikethroughButton:  { iconCss: "e-icons e-strikethrough", content:
"Strikethrough", isToggle: "true" },
        caseButton:  { iconCss: "e-icons e-change-case", content: "Change
Case", isToggle: "true" },
        colorPicker: {value: "#123456" },
        editButton: { iconCss: "e-icons e-edit", content: "Editor" },
        tableSettings:{
          iconCss: "e-icons e-table", content: "Table", isDropDownButton:
true,
          items: [{ text: "Insert Table" }, { text: "Draw Table" }, { text:
"Convert Table" }, { text: "Excel SpreadSheet" }]
        },
        chartSettings: { iconCss: "e-icons e-chart", content: "Chart" },
        printSettings: { iconCss: "e-icons e-print-layout", content: "Print
Layout" },
        layoutSettings: { iconCss: "e-icons e-web-layout", content: "Web
Layout" },
        rulerSettings:  { label: "Ruler", checked: false },
        gridSettings:  { label: "Gridlines", checked: false },
        navigationSettings:  { label: "Navigation Pane", checked: true },
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

  .ribbonTemplate {
    display: flex;
    align-items: center;
    justify-content: center;
    cursor: pointer;
  }
  .ribbonTemplate.Large {
    flex-direction: column;
  }
  .ribbonTemplate.Large .e-icons {
    font-size: 35px;
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  .ribbonTemplate.Medium .e-icons,
  .ribbonTemplate.Small .e-icons{
    font-size: 20px;
    margin: 15px 5px;
  }
  .ribbonTemplate.Small .text {
    display:none;
  }
  .font-group .e-ribbon-group-content {
    justify-content: center;
  }
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/ribbon/getting-started-cs1" %}

Getting Started with Syncfusion Ribbon Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Ribbon component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

Copyright © 2001 -2024 Syncfusion Inc.

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Ribbon component as an example. To use the Vue Ribbon component in the project, the @syncfusion/ej2-vue-ribbon package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-ribbon --save

`

or

`bash

yarn add @syncfusion/ej2-vue-ribbon

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Button component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Add Syncfusion Vue component

Follow the below steps to add the Vue Ribbon component using Composition API or Options API:

Importing the component, directives and required modules

First, import and register the Ribbon component, Ribbon directives and inject the required modules in the <script> section of the **src/App.vue** file.If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
//Component, Directives registeration
import { RibbonComponent as EjsRibbon, RibbonGroupDirective as ERibbonGroup,
RibbonGroupsDirective as ERibbonGroups, RibbonCollectionsDirective as
ERibbonCollections, RibbonCollectionDirective as ERibbonCollection,
RibbonItemsDirective as ERibbonItems, RibbonItemDirective as ERibbonItems,
RibbonTabsDirective as ERibbonTabs, RibbonTabDirective as ERibbonTab,
RibbonFileMenu, RibbonColorPicker } from "@syncfusion/ej2-vue-ribbon";
import { provide } from "vue";
const ribbon = [RibbonFileMenu, RibbonColorPicker];
provide('ribbon', ribbon);
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { RibbonComponent, RibbonGroupDirective, RibbonGroupsDirective,
RibbonCollectionsDirective, RibbonCollectionDirective, RibbonItemsDirective,
RibbonItemDirective, RibbonTabsDirective, RibbonTabDirective,RibbonFileMenu,
RibbonColorPicker } from "@syncfusion/ej2-vue-ribbon";
//Component, Directives registeration
export default {
components: {
'ejs-ribbon' : RibbonComponent,
'e-ribbon-tabs': RibbonTabsDirective,
'e-ribbon-tab': RibbonTabDirective,
'e-ribbon-groups': RibbonGroupsDirective,
'e-ribbon-group': RibbonGroupDirective,
'e-ribbon-collections': RibbonCollectionsDirective,
'e-ribbon-collection': RibbonCollectionDirective,
'e-ribbon-items': RibbonItemsDirective,
'e-ribbon-item': RibbonItemDirective
},
provide: {
ribbon: [RibbonFileMenu, RibbonColorPicker]
}
}
</script>
```

Adding the Ribbon component

Add the component definition in template section.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
```

```
<ejs-ribbon id="ribbon"></ejs-ribbon>
</template>
<script setup>
import { RibbonComponent as EjsRibbon} from "@syncfusion/ej2-vue-ribbon";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-ribbon id="ribbon"></ejs-ribbon>
</template>
<script>
import { RibbonComponent } from "@syncfusion/ej2-vue-ribbon";
export default {
components: {
"ejs-ribbon": RibbonComponent
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

Adding Ribbon Tab

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-ribbon id="ribbon">
<e-ribbon-tabs>
<e-ribbon-tab header="Home"></e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script setup>
import { RibbonComponent as EjsRibbon, RibbonTabsDirective as ERibbonTabs,
RibbonTabDirective as ERibbonTab } from "@syncfusion/ej2-vue-ribbon";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-ribbon id="ribbon">
<e-ribbon-tabs>
<e-ribbon-tab header="Home"></e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import { RibbonComponent, RibbonTabsDirective, RibbonTabDirective } from
"@syncfusion/ej2-vue-ribbon";
export default {
components: {
"ejs-ribbon": RibbonComponent,
'e-ribbon-tabs': RibbonTabsDirective,
'e-ribbon-tab': RibbonTabDirective
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

Adding Ribbon Group

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-ribbon id="ribbon">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard" orientation="Row"></e-ribbon-group>
</e-ribbon-groups>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script setup>
import { RibbonComponent as EjsRibbon, RibbonTabsDirective as EjsTabs,
RibbonTabDirective as EjsTab, RibbonGroupDirective as EjsRibbonGroup,
RibbonGroupsDirective as EjsRibbonGroups } from "@syncfusion/ej2-vue-
ribbon";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-ribbon id="ribbon">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard" orientation="Row"></e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import { RibbonComponent, RibbonTabsDirective, RibbonTabDirective,
RibbonGroupDirective, RibbonGroupsDirective } from "@syncfusion/ej2-vue-
ribbon";
export default {
components: {
"ejs-ribbon": RibbonComponent,
'e-ribbon-tabs': RibbonTabsDirective,
'e-ribbon-tab': RibbonTabDirective,
'e-ribbon-groups': RibbonGroupsDirective,
'e-ribbon-group': RibbonGroupDirective
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

The orientation property of ribbon group defines whether the collection of items will be rendered column-wise or row-wise.

Adding Ribbon Items

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-ribbon id="ribbon">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard" orientation="Column">
<e-ribbon-collections>
<e-ribbon-collection id="paste-collection">
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :splitButtonSettings="pasteSettings"></e-
ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
<e-ribbon-collection id="cutcopy-collection">
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton"></e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="copyButton"></e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script setup>
import { RibbonComponent as EjsRibbon, RibbonTabsDirective as ERibbonTabs,
RibbonTabDirective as ERibbonTab, RibbonGroupDirective as ERibbonGroup,
RibbonGroupsDirective as ERibbonGroups, RibbonCollectionsDirective as
ERibbonCollections, RibbonCollectionDirective as ERibbonCollection,
RibbonItemsDirective as ERibbonItems, RibbonItemDirective as ERibbonItem }
from "@syncfusion/ej2-vue-ribbon";
const pasteSettings = {
iconCss: "e-icons e-paste", content: "Paste",
items: [{ text: "Keep Source Format" },
{ text: "Merge format" },
{ text: "Keep text only" }]
}
const cutButton = { iconCss: "e-icons e-cut", content: "Cut" }
const copyButton = { iconCss: "e-icons e-copy", content: "Copy" }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-ribbon id="ribbon">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard" orientation="Column">
<e-ribbon-collections>
<e-ribbon-collection id="paste-collection">
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :splitButtonSettings="pasteSettings"></e-
ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
<e-ribbon-collection id="cutcopy-collection">
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton"></e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="copyButton"></e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import { RibbonComponent, RibbonTabsDirective, RibbonTabDirective,
RibbonGroupDirective, RibbonGroupsDirective, RibbonCollectionsDirective,
RibbonCollectionDirective, RibbonItemsDirective, RibbonItemDirective } from
"@syncfusion/ej2-vue-ribbon";
export default {
components: {
"ejs-ribbon": RibbonComponent,
'e-ribbon-tabs': RibbonTabsDirective,
'e-ribbon-tab': RibbonTabDirective,
'e-ribbon-groups': RibbonGroupsDirective,
'e-ribbon-group': RibbonGroupDirective,
'e-ribbon-collections': RibbonCollectionsDirective,
'e-ribbon-collection': RibbonCollectionDirective,
'e-ribbon-items': RibbonItemsDirective,
'e-ribbon-item': RibbonItemDirective
},
data: function () {
return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
pasteSettings:{
iconCss: "e-icons e-paste", content: "Paste",
items: [{ text: "Keep Source Format" }, { text: "Merge format" }, { text:
"Keep text only" }]
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
};
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

The type property to specify the type of component to be rendered, like a button, a drop-down button, a combo box, and more.

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-ribbon :fileMenu="fileSettings">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard" groupIconCss="e-icons e-paste"
:showLauncherIcon="true" >
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :allowedSizes="largeSize"
:splitButtonSettings="pasteSettings" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton">
</e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="copyButton">
</e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="formatButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-group>
<e-ribbon-group header="Font" orientation="Row" :enableGroupOverflow="true"
:isCollapsible="false" groupIconCss="e-icons e-bold" cssClass="font-group" >
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ComboBox" :comboBoxSettings="styleOptions" >
</e-ribbon-item>
<e-ribbon-item type="ComboBox" :comboBoxSettings="sizeOptions">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ColorPicker" displayOptions="Simplified"
:allowedSizes="smallSize" :colorPickerSettings="colorPicker" >
</e-ribbon-item>
<e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="boldButton" >
</e-ribbon-item>
<e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="italicButton">
</e-ribbon-item>
<e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="underlineButton" >
</e-ribbon-item>
<e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="strikethroughButton">
</e-ribbon-item>
<e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="caseButton" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
<e-ribbon-group header="Editor" :isCollapsible="false" groupIconCss="e-icons
e-edit" >
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :allowedSizes="largeSize"
:buttonSettings="editButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
<e-ribbon-tab header="Insert">
<e-ribbon-groups>
<e-ribbon-group header="Tables" :isCollapsible=false>
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-item type="DropDown" :allowedSizes="largeSize"
:dropDownSettings="tableSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
<e-ribbon-group header="Illustrations" id="illustration" orientation="Row"
:enableGroupOverflow=true groupIconCss="e-icons e-image">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="chartSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
<e-ribbon-group header="Media" :isCollapsible=false>
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Template" :itemTemplate="'ribbonTemplate'">
<template v-slot:ribbonTemplate = "{data}">
<span v-bind:class="'ribbonTemplate ' + data.activeSize"><span class="e-
icons e-video"></span><span class="text">Video</span></span>
</template>
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
<e-ribbon-tab header="View">
<e-ribbon-groups>
<e-ribbon-group header="Views" orientation="Row" groupIconCss="e-icons e-
print">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="printSettings">
</e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="layoutSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
<e-ribbon-group header="show" :isCollapsible=false>
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="CheckBox" :checkBoxSettings="rulerSettings">
</e-ribbon-item>
<e-ribbon-item type="CheckBox" :checkBoxSettings="gridSettings">
</e-ribbon-item>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-item type="CheckBox" :checkBoxSettings="navigationSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script setup>
import { RibbonComponent as EjsRibbon, RibbonFileMenu, RibbonItemSize,
RibbonColorPicker, RibbonGroupDirective as ERibbonGroup,
RibbonGroupsDirective as ERibbonGroups, RibbonCollectionsDirective as
ERibbonCollections, RibbonCollectionDirective as ERibbonCollection,
RibbonItemsDirective as ERibbonItems, RibbonItemDirective as ERibbonItem,
RibbonTabsDirective as ERibbonTabs, RibbonTabDirective as ERibbonTab} from
"@syncfusion/ej2-vue-ribbon";
import { provide } from "vue";
const ribbon = [RibbonFileMenu, RibbonColorPicker];
provide('ribbon', ribbon);
const largeSize = RibbonItemSize.Large;
const smallSize = RibbonItemSize.Small;
const fileSettings = {
visible: true,
menuItems: [
{ text: "New", iconCss: "e-icons e-file-new", id: "new" },
{ text: "Open", iconCss: "e-icons e-folder-open", id: "open" },
{ text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
{ text: "Save as", iconCss: "e-icons e-save", id: "save" }
]
};
const pasteSettings = {
iconCss: "e-icons e-paste", content: "Paste",
items: [{ text: "Keep Source Format" }, { text: "Merge format" }, { text:
"Keep text only" }]
};
const cutButton = { iconCss: "e-icons e-cut", content: "Cut" };
const copyButton = { iconCss: "e-icons e-copy", content: "Copy" };
const formatButton = { iconCss: "e-icons e-format-painter", content: "Format
Painter" };
const styleOptions = {
dataSource: ["Algerian", "Arial", "Calibri", "Cambria", "Cambria Math",
"Courier New", "Candara", "Georgia", "Impact", "Segoe Print", "Segoe
Script", "Segoe UI", "Symbol", "Times New Roman", "Verdana", "Windings" ],
index: 3,
width: "150px",
allowFiltering: true
};
const sizeOptions = {
dataSource: ["8", "9", "10", "11", "12", "14", "16", "18", "20", "22", "24",
"26", "28", "36", "48", "72", "96" ],
index: 4,
width: "65px"
};
```

Copyright © 2001 -2024 Syncfusion Inc.

```
const boldButton = { iconCss: "e-icons e-bold", content: "Bold", isToggle:
"true" };
const italicButton = { iconCss: "e-icons e-italic", content: "Italic",
isToggle: "true" };
const underlineButton = { iconCss: "e-icons e-underline", content:
"Underline", isToggle: "true" };
const strikethroughButton = { iconCss: "e-icons e-strikethrough", content:
"Strikethrough", isToggle: "true" };
const caseButton = { iconCss: "e-icons e-change-case", content: "Change
Case", isToggle: "true" };
const colorPicker = { value: "#123456" };
const editButton = { iconCss: "e-icons e-edit", content: "Editor" };
const tableSettings = {
iconCss: "e-icons e-table", content: "Table", isDropDownButton: true,
items: [{ text: "Insert Table" }, { text: "Draw Table" }, { text: "Convert
Table" }, { text: "Excel SpreadSheet" }]
};
const chartSettings = { iconCss: "e-icons e-chart", content: "Chart" };
const  printSettings = { iconCss: "e-icons e-print-layout", content: "Print
Layout" };
const  layoutSettings = { iconCss: "e-icons e-web-layout", content: "Web
Layout" };
const  rulerSettings =  { label: "Ruler", checked: false };
const  gridSettings =  { label: "Gridlines", checked: false };
const  navigationSettings =  { label: "Navigation Pane", checked: true };
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
.ribbonTemplate {
display: flex;
align-items: center;
justify-content: center;
cursor: pointer;
}
.ribbonTemplate.Large {
flex-direction: column;
}
.ribbonTemplate.Large .e-icons {
font-size: 35px;
}
.ribbonTemplate.Medium .e-icons,
.ribbonTemplate.Small .e-icons{
font-size: 20px;
margin: 15px 5px;
}
.ribbonTemplate.Small .text {
display:none;
}
.font-group .e-ribbon-group-content {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
justify-content: center;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-ribbon :fileMenu="fileSettings">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard" groupIconCss="e-icons e-paste"
:showLauncherIcon="true" >
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :allowedSizes="largeSize"
:splitButtonSettings="pasteSettings" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton">
</e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="copyButton">
</e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="formatButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
<e-ribbon-group header="Font" orientation="Row" :enableGroupOverflow="true"
:isCollapsible="false" groupIconCss="e-icons e-bold" cssClass="font-group" >
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ComboBox" :comboBoxSettings="styleOptions" >
</e-ribbon-item>
<e-ribbon-item type="ComboBox" :comboBoxSettings="sizeOptions">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ColorPicker" displayOptions="Simplified"
:allowedSizes="smallSize" :colorPickerSettings="colorPicker" >
</e-ribbon-item>
<e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="boldButton" >
</e-ribbon-item>
<e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="italicButton">
</e-ribbon-item>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="underlineButton" >
</e-ribbon-item>
<e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="strikethroughButton">
</e-ribbon-item>
<e-ribbon-item type="Button" :allowedSizes="smallSize"
:buttonSettings="caseButton" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
<e-ribbon-group header="Editor" :isCollapsible="false" groupIconCss="e-icons
e-edit" >
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :allowedSizes="largeSize"
:buttonSettings="editButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
<e-ribbon-tab header="Insert">
<e-ribbon-groups>
<e-ribbon-group header="Tables" :isCollapsible=false>
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="DropDown" :allowedSizes="largeSize"
:dropDownSettings="tableSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
<e-ribbon-group header="Illustrations" id="illustration" orientation="Row"
:enableGroupOverflow=true groupIconCss="e-icons e-image">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="chartSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
<e-ribbon-group header="Media" :isCollapsible=false>
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Template" :itemTemplate="'ribbonTemplate'">
<template v-slot:ribbonTemplate = "{data}">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<span v-bind:class="'ribbonTemplate ' + data.activeSize"><span class="e-
icons e-video"></span><span class="text">Video</span></span>
</template>
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
<e-ribbon-tab header="View">
<e-ribbon-groups>
<e-ribbon-group header="Views" orientation="Row" groupIconCss="e-icons e-
print">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="printSettings">
</e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="layoutSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
<e-ribbon-group header="show" :isCollapsible=false>
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="CheckBox" :checkBoxSettings="rulerSettings">
</e-ribbon-item>
<e-ribbon-item type="CheckBox" :checkBoxSettings="gridSettings">
</e-ribbon-item>
<e-ribbon-item type="CheckBox" :checkBoxSettings="navigationSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import { RibbonComponent, RibbonFileMenu, RibbonItemSize, RibbonColorPicker,
RibbonGroupDirective, RibbonGroupsDirective, RibbonCollectionsDirective,
RibbonCollectionDirective, RibbonItemsDirective, RibbonItemDirective,
RibbonTabsDirective, RibbonTabDirective } from "@syncfusion/ej2-vue-ribbon";
export default {
components: {
"ejs-ribbon": RibbonComponent,
'e-ribbon-tabs': RibbonTabsDirective,
'e-ribbon-tab': RibbonTabDirective,
'e-ribbon-groups': RibbonGroupsDirective,
'e-ribbon-group': RibbonGroupDirective,
'e-ribbon-collections': RibbonCollectionsDirective,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
'e-ribbon-collection': RibbonCollectionDirective,
'e-ribbon-items': RibbonItemsDirective,
'e-ribbon-item': RibbonItemDirective
},
provide: {
ribbon: [RibbonFileMenu, RibbonColorPicker]
},
data: function () {
return {
largeSize: RibbonItemSize.Large,
smallSize: RibbonItemSize.Small,
fileSettings: {
visible: true,
menuItems: [
{ text: "New", iconCss: "e-icons e-file-new", id: "new" },
{ text: "Open", iconCss: "e-icons e-folder-open", id: "open" },
{ text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
{ text: "Save as", iconCss: "e-icons e-save", id: "save" }
]
},
pasteSettings:{
iconCss: "e-icons e-paste", content: "Paste",
items: [{ text: "Keep Source Format" }, { text: "Merge format" }, { text:
"Keep text only" }]
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
formatButton:  { iconCss: "e-icons e-format-painter", content: "Format
Painter" },
styleOptions: {
dataSource: ["Algerian", "Arial", "Calibri", "Cambria", "Cambria Math",
"Courier New", "Candara", "Georgia", "Impact", "Segoe Print", "Segoe
Script", "Segoe UI", "Symbol", "Times New Roman", "Verdana", "Windings" ],
index: 3,
width: "150px",
allowFiltering: true
},
sizeOptions: {
dataSource: ["8", "9", "10", "11", "12", "14", "16", "18", "20", "22", "24",
"26", "28", "36", "48", "72", "96" ],
index: 4,
width: "65px"
},
boldButton:  { iconCss: "e-icons e-bold", content: "Bold", isToggle: "true"
},
italicButton:  { iconCss: "e-icons e-italic", content: "Italic", isToggle:
"true" },
underlineButton:  { iconCss: "e-icons e-underline", content: "Underline",
isToggle: "true" },
strikethroughButton:  { iconCss: "e-icons e-strikethrough", content:
"Strikethrough", isToggle: "true" },
caseButton:  { iconCss: "e-icons e-change-case", content: "Change Case",
isToggle: "true" },
colorPicker: {value: "#123456" },
editButton: { iconCss: "e-icons e-edit", content: "Editor" },
tableSettings:{
iconCss: "e-icons e-table", content: "Table", isDropDownButton: true,
```

```
items: [{ text: "Insert Table" }, { text: "Draw Table" }, { text: "Convert
Table" }, { text: "Excel SpreadSheet" }]
},
chartSettings: { iconCss: "e-icons e-chart", content: "Chart" },
printSettings: { iconCss: "e-icons e-print-layout", content: "Print Layout"
},
layoutSettings: { iconCss: "e-icons e-web-layout", content: "Web Layout" },
rulerSettings:  { label: "Ruler", checked: false },
gridSettings:  { label: "Gridlines", checked: false },
navigationSettings:  { label: "Navigation Pane", checked: true },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
.ribbonTemplate {
display: flex;
align-items: center;
justify-content: center;
cursor: pointer;
}
.ribbonTemplate.Large {
flex-direction: column;
}
.ribbonTemplate.Large .e-icons {
font-size: 35px;
}
.ribbonTemplate.Medium .e-icons,
.ribbonTemplate.Small .e-icons{
font-size: 20px;
margin: 15px 5px;
}
.ribbonTemplate.Small .text {
display:none;
}
.font-group .e-ribbon-group-content {
justify-content: center;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

Copyright © 2001 -2024 Syncfusion Inc.

`

or

`bash

yarn run dev

`

The output will appear as follows:

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Modules in Ribbon component

The following modules are available in Ribbon. If the module injection type is **selective**, manual injection is required to extend the Ribbon's functionality.

| Module | Description | Module Injection Type |

|------|-------------|------|

| RibbonButton | To use the built-in button as a ribbon item. | default |

| RibbonCheckBox | To use the built-in checkbox as a ribbon item.| default |

| RibbonDropDown | To use the built-in dropdown button as a ribbon item.| default |

| RibbonSplitButton | To use the built-in split button as a ribbon item.| default |

| RibbonComboBox | To use the built-in combobox as a ribbon item. | default |

| RibbonColorPicker | Inject this module to use the built-in colorpicker as a ribbon item.| selective |

| RibbonFileMenu | Inject this module to use the file menu feature.| selective |

These selective modules should be injected into the provide section and use ribbon as a key of the object.

`html

<template>

<ejs-ribbon>

// Render Tabs here

Copyright © 2001 -2024 Syncfusion Inc.

</ejs-ribbon>

</template>

<script>

import { RibbonFileMenu, RibbonColorPicker } from "@syncfusion/ej2-vue-ribbon";

export default {

provide: {

ribbon: [RibbonFileMenu, RibbonColorPicker]

},

};

</script>

`

Tabs and Groups

The Ribbon component consists of a series of tabs that are organized into groups to enable quick access to specific commands or tools. Each tab contains a set of groups, and each group contains collections of items that are logically related to each other.

Adding Tabs

You can use the tabs property to add tabs to the Ribbon component and define the content of the tab header by using the header property.

APP.VUE

```
<template>
  <ejs-ribbon>
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
      </e-ribbon-tab>
      <e-ribbon-tab header="Insert">
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/tabs" %}

Copyright © 2001 -2024 Syncfusion Inc.

Adding Groups

You can use the groups property to add groups for each tab in the Ribbon and define the name of the group header by using the header property.

APP.VUE

```
<template>
  <ejs-ribbon>
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Clipboard">
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
      <e-ribbon-tab header="Insert">
        <e-ribbon-groups>
          <e-ribbon-group header="Tables">
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/groups" %}

Adding Items

You can add collections of items to each group by using the collections and items properties.

APP.VUE

```
<template>
  <ejs-ribbon>
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Clipboard">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="SplitButton"
:allowedSizes="largeSize" :splitButtonSettings="pasteSettigs" >
                  </e-ribbon-item>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
      <e-ribbon-tab header="Insert">
        <e-ribbon-groups>
          <e-ribbon-group header="Tables">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button" :buttonSettings="copyButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button"
:buttonSettings="formatButton">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonItemSize, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        largeSize: RibbonItemSize.Large,
        pasteSettigs:{
          iconCss: "e-icons e-paste", content: "Paste",
          items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
        },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter" }
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/items" %}

For more information on the built-in and how to add custom Ribbon items, you can visit the items page.

Ribbon Items

Ribbon renders various built-in items based on the item type property. By default, the type property is set as `Button` which renders the Button.

Built-in items

You can render the built-in Ribbon items by using the `<e-ribbon-item>` tag directive, to specify the type property.

The following table explains the built-in items and their actions.

| Built-in Ribbon Items | Actions |

|---|---|

| Button | Renders button as ribbon item.|

| CheckBox | Renders checkbox as ribbon item.|

| DropDown | Renders dropdownbutton as ribbon item.|

| SplitButton | Renders splitbutton as ribbon item.|

| ComboBox | Renders combobox as ribbon item.|

| ColorPicker | Renders color picker as ribbon item.|

| GroupButton | Renders groupbutton as ribbon item.|

Button items

You can render the built-in button Ribbon item by setting the type property as `Button`. You can also customize the button item using the RibbonButtonSettingsModel, which provides options such as `iconCss`, `content`, `isToggle` and more.

Toggle button

The isToggle property can be used to define whether the button act as a toggle button or not. By default, the value is `false`.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Clipboard">

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
cutButton:  { iconCss: "e-icons e-cut", content: "Cut", isToggle: true },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

Checkbox items

You can render the built-in checkBox Ribbon item by setting the type property to CheckBox. You can also customize the checkBox item using the RibbonCheckBoxSettingsModel, which provides options such as labelPosition, label, checked and more.

Checkbox state

You can use the checked property to handle the checked or unchecked state. By default, the value is false.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Clipboard">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="CheckBox" :checkBoxSettings="rulerSettings">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

<script>

import Vue from "vue";

Copyright © 2001 -2024 Syncfusion Inc.

```
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
rulerSettings: {
label: 'Ruler',
checked: true
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

Defining label

You can use the label property to add a caption for the CheckBox. The label position can be set Before or After, by using the labelPosition property. By default, the labelPosition is After.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
```

```
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="CheckBox" :checkBoxSettings="rulerSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
rulerSettings: {
label: 'Ruler',
checked: true,
labelPosition: "Before"
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";

@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

DropDown button items

You can render the built-in dropDown Ribbon item by setting the type property to DropDown. You can also customize the dropDown item through RibbonDropDownSettingsModel, which provides options such as iconCss, content, target and more.

Target

The target property specifies the element selector to be displayed in the DropDownButton popup.

APP.VUE

```
<template>
  <div>
    <ejs-ribbon>
      <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
          <e-ribbon-groups>
            <e-ribbon-group header="Table">
              <e-ribbon-collections>
                <e-ribbon-collection>
                  <e-ribbon-items>
                    <e-ribbon-item type="DropDown"
:dropDownSettings="tableSettings">
                    </e-ribbon-item>
                  </e-ribbon-items>
                </e-ribbon-collection>
              </e-ribbon-collections>
            </e-ribbon-group>
          </e-ribbon-groups>
        </e-ribbon-tab>
      </e-ribbon-tabs>
    </ejs-ribbon>
    <ejs-listview id='tableList' :dataSource='tableOptions'
headerTitle='Table' showHeader='true'></ejs-listview>
  </div>
</template>
<script>
  import Vue from "vue";
  import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  import { ListViewPlugin } from '@syncfusion/ej2-vue-lists';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  Vue.use(RibbonPlugin);
  Vue.use(ListViewPlugin );
  export default {
    data: function () {
    return {
      tableOptions: [
        { text: "Insert Table" },
        { text: "This device" },
        { text: "Convert Table" },
        { text: "Excel SpreadSheet" }
      ],
      tableSettings: {
        content: 'Table',
        iconCss: 'e-icons e-table',
        target: "#tableList"
      }
    };
  }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/dropdown-target" %}

Split button items

You can render the built-in splitButton Ribbon item by setting the type property to SplitButton. You can also customize the splitButton item through RibbonSplitButtonSettingsModel, which provides options such as iconCss, items, target and more.

Target

The target property specifies the element selector to be displayed in the SplitButton popup.

APP.VUE

```
<template>
  <div>
    <ejs-ribbon>
      <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
          <e-ribbon-groups>
            <e-ribbon-group header="Table">
              <e-ribbon-collections>
                <e-ribbon-collection>
                  <e-ribbon-items>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <e-ribbon-item type="SplitButton"
:splitButtonSettings="tableSettings">
                    </e-ribbon-item>
                  </e-ribbon-items>
                </e-ribbon-collection>
              </e-ribbon-collections>
            </e-ribbon-group>
          </e-ribbon-groups>
        </e-ribbon-tab>
      </e-ribbon-tabs>
    </ejs-ribbon>
    <ejs-listview id='tableList' :dataSource='tableOptions'
headerTitle='Table' showHeader='true'></ejs-listview>
  </div>
</template>
<script>
  import Vue from "vue";
  import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  import { ListViewPlugin } from '@syncfusion/ej2-vue-lists';
  Vue.use(RibbonPlugin);
  Vue.use(ListViewPlugin );
  export default {
    data: function () {
    return {
      tableOptions: [
        { text: "Insert Table" },
        { text: "This device" },
        { text: "Convert Table" },
        { text: "Excel SpreadSheet" }
      ],
      tableSettings: {
        content: 'Table',
        iconCss: 'e-icons e-table',
        target: "#tableList"
      }
    };
  }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/splitbutton-target" %}

Copyright © 2001 -2024 Syncfusion Inc.

Combobox items

You can render the built-in comboBox Ribbon item by setting the type property to ComboBox. You can also customize the comboBox item through RibbonComboBoxSettingsModel, which provides options such as allowFiltering, autoFill, index, sortOrder and more.

Filtering

You can use the allowFiltering property to filter the data items. The filtering operation is initiated automatically, as soon as you start typing characters. If no match is found the value of the noRecordsTemplate property will be displayed. By default, the value is false.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ComboBox" :comboBoxSettings="styleOptions" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
return {
styleOptions: {
dataSource: ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math', 'Courier New', 'Candara', 'Georgia'],
allowFiltering: true
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

Index

You can use the index property to get or set the selected item in the combobox.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-item type="ComboBox" :comboBoxSettings="styleOptions" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
styleOptions: {
dataSource: ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math', 'Courier New', 'Candara', 'Georgia'],
index: 3
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

SortOrder

You can use the sortOrder property to specify the order in which the DataSource should be sorted.

None	The data source is not sorted.
Ascending	The data source is sorted in ascending order.
Descending	The data source is sorted in descending order.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Font">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="ComboBox" :comboBoxSettings="styleOptions" >

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

<script>

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
styleOptions: {
dataSource: ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math', 'Courier New', 'Candara', 'Georgia'],
index: 3,
sortOrder: "Descending"
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

Colorpicker items

You can render the built-in colorPicker Ribbon item by setting the type property to `ColorPicker`. You can also customize the colorPicker item through RibbonColorPickerSettingsModel, which provides options such as `value`, `columns`, `showButtons` and more.

Value

You can use the value property to specify the color value. The value should be specified as Hex code.

```
`html
```

```
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ColorPicker" :colorPickerSettings="colorPicker" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonColorPicker } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [ RibbonColorPicker ]
},
data: function () {
return {
colorPicker: {
value: "#123456"
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

Groupbutton items

You can render the built-in groupbutton Ribbon item by setting the type property to `GroupButton`. You can also customize the groupbutton item using the `RibbonGroupButtonSettingsModel`, which provides options such as `selection` and `items`.

Items

You can render the groupbutton items by using items property. You can also customize the groupbutton items through RibbonGroupButtonItemModel, which provides options such as content, iconCss, selected and more.

Item content

You can use the content property to define the text content for the groupbutton.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Paragraph">
<e-ribbon-collections>
<e-ribbon-collection>
```

```
<e-ribbon-items>
<e-ribbon-item type="GroupButton" :allowedSizes="size"
:groupButtonSettings="groupButtonItem"></e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonItemSize, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
size: RibbonItemSize.Medium,
groupButtonItem: {
items: [
{iconCss: 'e-icons e-align-left', content: 'Align Left'},
{iconCss: 'e-icons e-align-center',content: 'Align Center'},
{iconCss: 'e-icons e-align-right',content: 'Align Right'},
{iconCss: 'e-icons e-justify',content: 'Justify'}
]
},
};
}
};
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-base/styles/material.css";

@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";

@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

Icon only

You can use the iconCss property to customize the groupbutton icon. If the iconCss property is not defined, the groupbutton will not be rendered.

APP.VUE

```
<template>
  <ejs-ribbon>
  <e-ribbon-tabs>
    <e-ribbon-tab header="Home">
      <e-ribbon-groups>
        <e-ribbon-group header="Paragraph">
          <e-ribbon-collections>
            <e-ribbon-collection>
              <e-ribbon-items>
                <e-ribbon-item type="GroupButton" :allowedSizes="size"
:groupButtonSettings="groupButtonIcon"></e-ribbon-item>
              </e-ribbon-items>
            </e-ribbon-collection>
          </e-ribbon-collections>
        </e-ribbon-group>
      </e-ribbon-groups>
    </e-ribbon-tab>
  </e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonItemSize, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
  data: function () {
    return {
      size: RibbonItemSize.Small,
      groupButtonIcon: {
        items: [
          {iconCss: 'e-icons e-align-left'},
          {iconCss: 'e-icons e-align-center'},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            {iconCss: 'e-icons e-align-right'},
            {iconCss: 'e-icons e-justify'}
          ]
        },
      };
    }
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/groupButtonIcon" %}

Selection

You can use the selected property to select the groupbutton item initally. When set to true, the button will be selected. By default the selected property is false.

APP.VUE

```
<template>
  <ejs-ribbon>
  <e-ribbon-tabs>
    <e-ribbon-tab header="Home">
      <e-ribbon-groups>
        <e-ribbon-group header="Paragraph">
          <e-ribbon-collections>
            <e-ribbon-collection>
              <e-ribbon-items>
                <e-ribbon-item type="GroupButton" :allowedSizes="size"
:groupButtonSettings="groupButtonSelected"></e-ribbon-item>
              </e-ribbon-items>
            </e-ribbon-collection>
          </e-ribbon-collections>
        </e-ribbon-group>
      </e-ribbon-groups>
    </e-ribbon-tab>
  </e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonItemSize, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
  data: function () {
    return {
      size: RibbonItemSize.Medium,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      groupButtonSelected: {
        items: [
          {iconCss: 'e-icons e-align-left', selected: true, content: 'Align
Left'},
          {iconCss: 'e-icons e-align-center',content: 'Align Center'},
          {iconCss: 'e-icons e-align-right',content: 'Align Right'},
          {iconCss: 'e-icons e-justify',content: 'Justify'}
        ]
      },
    };
  }
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/groupButtonSelected" %}

Single selection

You can set the selection property value as `RibbonGroupButtonSelection.Single` to make one selection at a time. It automatically deselects the previous choice when a different item is clicked.

APP.VUE

```
<template>
  <ejs-ribbon>
  <e-ribbon-tabs>
    <e-ribbon-tab header="Home">
      <e-ribbon-groups>
        <e-ribbon-group header="Paragraph">
          <e-ribbon-collections>
            <e-ribbon-collection>
              <e-ribbon-items>
                <e-ribbon-item type="GroupButton" :allowedSizes="size"
:groupButtonSettings="groupButtonSingle"></e-ribbon-item>
              </e-ribbon-items>
            </e-ribbon-collection>
          </e-ribbon-collections>
        </e-ribbon-group>
      </e-ribbon-groups>
    </e-ribbon-tab>
  </e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { RibbonItemSize, RibbonGroupButtonSelection, RibbonPlugin } from
"@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
  data: function () {
    return {
      size: RibbonItemSize.Medium,
        groupButtonSingle: {
        selection: RibbonGroupButtonSelection.Single,
        items: [
          {iconCss: 'e-icons e-align-left', selected: true, content: 'Align
Left'},
          {iconCss: 'e-icons e-align-center',content: 'Align Center'},
          {iconCss: 'e-icons e-align-right',content: 'Align Right'},
          {iconCss: 'e-icons e-justify',content: 'Justify'}
        ]
      },
    };
  }
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/singleSelection" %}

Multiple selection

You can set the selection property value as `RibbonGroupButtonSelection.Multiple` to select more than one button at a time. Users can select a button one by one to select multiple buttons.

APP.VUE

```
<template>
  <ejs-ribbon>
  <e-ribbon-tabs>
    <e-ribbon-tab header="Home">
      <e-ribbon-groups>
        <e-ribbon-group header="Paragraph">
          <e-ribbon-collections>
            <e-ribbon-collection>
              <e-ribbon-items>
                <e-ribbon-item type="GroupButton" :allowedSizes="size"
:groupButtonSettings="groupButtonMultiple"></e-ribbon-item>
              </e-ribbon-items>
            </e-ribbon-collection>
          </e-ribbon-collections>
        </e-ribbon-group>
```

```
      </e-ribbon-groups>
    </e-ribbon-tab>
  </e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonItemSize, RibbonGroupButtonSelection, RibbonPlugin } from
"@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
  data: function () {
    return {
      size: RibbonItemSize.Medium,
      groupButtonMultiple: {
            selection: RibbonGroupButtonSelection.Multiple,
            items: [{ iconCss: 'e-icons e-bold', content: 'Bold', selected:
true},
            {iconCss: 'e-icons e-italic', content: 'Italic'},
            {iconCss: 'e-icons e-underline', content: 'Underline'},
            {iconCss: 'e-icons e-strikethrough', content: 'Strikethrough'},
            {iconCss: 'e-icons e-change-case', content: 'Change Case'}
          ]
        }
    };
  }
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/multipleSelection" %}

Groupbutton in simplified mode layout

In simplified mode, the groupbutton will be rendered as a dropdownbutton. The dropdownbutton icon will be updated based on the button item selected. The initial button icon will be the set, if none of the buttons are selected.

APP.VUE

```
<template>
  <ejs-ribbon activeLayout="Simplified">
  <e-ribbon-tabs>
    <e-ribbon-tab header="Home">
      <e-ribbon-groups>
        <e-ribbon-group header="Paragraph">
          <e-ribbon-collections>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <e-ribbon-collection>
              <e-ribbon-items>
                <e-ribbon-item type="GroupButton" :allowedSizes="size"
:groupButtonSettings="groupButtonSingle"></e-ribbon-item>
              </e-ribbon-items>
            </e-ribbon-collection>
          </e-ribbon-collections>
        </e-ribbon-group>
      </e-ribbon-groups>
    </e-ribbon-tab>
  </e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonItemSize, RibbonGroupButtonSelection, RibbonPlugin } from
"@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
  data: function () {
    return {
      size: RibbonItemSize.Medium,
        groupButtonSingle: {
        selection: RibbonGroupButtonSelection.Single,
        items: [
          {iconCss: 'e-icons e-align-left', selected: true, content: 'Align
Left'},
          {iconCss: 'e-icons e-align-center',content: 'Align Center'},
          {iconCss: 'e-icons e-align-right',content: 'Align Right'},
          {iconCss: 'e-icons e-justify',content: 'Justify'}
        ]
      },
    };
  }
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/simplifiedModeGroupButton" %}

Custom items

You can customize the ribbon items with non-built-in items or HTML content by setting the type property to `Template`. This provides an option to customize the ribbon items with greater flexibility.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <ejs-ribbon>
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Font">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Template"
:itemTemplate="'ribbonTemplate'">
                    <template v-slot:ribbonTemplate = "{data}">
                      <span v-bind:class="'ribbonTemplate ' +
data.activeSize"><span class="e-icons e-video"></span><span
class="text">Video</span></span>
                    </template>
                  </e-ribbon-item>
                  </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
  .ribbonTemplate {
    display: flex;
    align-items: center;
    justify-content: center;
    cursor: pointer;
  }
  .ribbonTemplate.Large {
    flex-direction: column;
  }
  .ribbonTemplate.Large .e-icons {
    font-size: 35px;
  }
  .ribbonTemplate.Medium .e-icons,
  .ribbonTemplate.Small .e-icons{
    font-size: 20px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    margin: 15px 5px;
  }
  .ribbonTemplate.Small .text {
    display:none;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/customItems" %}

Items display Mode

You can use the displayOptions property to display the items in the Ribbon layout.

Auto	The items are displayed in all layouts based on the ribbon's overflow state.
Classic	The items are displayed only in the classic layout group.
Simplified	The items are displayed only in the simplified layout group.
Overflow	The items are displayed only in the overflow popup.

Display items in Classic only

To diplay the items only in the classic layout group, set the mode as DisplayMode.Classic in the displayOptions property.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Clipboard">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="Button" :displayOptions="buttonDisplayMode" :buttonSettings="cutButton">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin,DisplayMode } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
buttonDisplayMode: DisplayMode.Classic,
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

Display items in Simplified only

To diplay the items only in the simplified layout group, set the mode as DisplayMode.Simplified in the displayOptions property.

```
`html
<template>
<ejs-ribbon>
```

```
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :displayOptions="buttonDisplayMode" :buttonSettings="cutButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin,DisplayMode } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
buttonDisplayMode: DisplayMode.Simplified,
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";

@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

Display items in Overflow popup only

To diplay the items only in the overflow, set the mode as DisplayMode.Overflow in the displayOptions property.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Clipboard">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="Button" :displayOptions="buttonDisplayMode" :buttonSettings="cutButton">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { RibbonPlugin,DisplayMode } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
buttonDisplayMode: DisplayMode.Overflow,
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

Enable or disable items

You can use the disabled property to disable a Ribbon item. It prevents the user interaction when set to `true`. By default, the value is `false`.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :disabled="true" :buttonSettings="cutButton"></e-ribbon-item>
<e-ribbon-item type="CheckBox" :disabled="true" :checkBoxSettings="rulerSettings"></e-ribbon-item>
<e-ribbon-item type="DropDown" :disabled="true" :dropDownSettings="tableSettings"></e-ribbon-item>
<e-ribbon-item type="SplitButton" :disabled="true" :splitButtonSettings="tableSettings"></e-ribbon-item>
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
rulerSettings: { label: 'Ruler', checked: true },
tableSettings: { content: 'Table', iconCss: 'e-icons e-table'}
};
}
};
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

<style>

@import "../node_modules/@syncfusion/ej2-base/styles/material.css";

@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";

@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

Ribbon Layouts

The Ribbon allows to customize the layout by using the activeLayout property. The Ribbon component supports the following layouts:

Classic layout

In classic layout, the Ribbon component organizes the items and groups in a traditional form by setting the activeLayout property to Classic. By default, the Ribbon component renders in the Classic layout.

APP.VUE

```
<template>
  <ejs-ribbon activeLayout="Classic">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Clipboard">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="SplitButton"
:allowedSizes="largeSize" :splitButtonSettings="pasteSettigs" >
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
      <e-ribbon-tab header="Insert">
        <e-ribbon-groups>
          <e-ribbon-group header="Tables">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <e-ribbon-item type="Button" :buttonSettings="copyButton">
                    </e-ribbon-item>
                    <e-ribbon-item type="Button"
:buttonSettings="formatButton">
                    </e-ribbon-item>
                  </e-ribbon-items>
                </e-ribbon-collection>
              </e-ribbon-collections>
            </e-ribbon-group>
          </e-ribbon-groups>
        </e-ribbon-tab>
      </e-ribbon-tabs>
    </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonItemSize, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        largeSize: RibbonItemSize.Large,
        pasteSettigs:{
          iconCss: "e-icons e-paste", content: "Paste",
          items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
        },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter" }
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/classic" %}

Defining items size

You can use the allowedSizes property to set the allowed size for an item. The Ribbon items can be appeared in three different sizes: Large(large icon with text), Medium(small icon with text) and Small(small icon only). On resizing, the items size can be changed based on the available width of the tab content from the order of Large-> Medium-> Small and viceversa.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <ejs-ribbon id="ribbon">
    <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
            <e-ribbon-groups>
                <e-ribbon-group>
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="SplitButton"
:allowedSizes="largeSize" :splitButtonSettings="pasteButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="Button"
:allowedSizes="mediumSize" :buttonSettings="cutButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:allowedSizes="smallSize" :buttonSettings="copyButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                    </e-ribbon-collections>
                </e-ribbon-group>
            </e-ribbon-groups>
        </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonItemSize, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        largeSize: RibbonItemSize.Large,
        mediumSize: RibbonItemSize.Medium,
        smallSize: RibbonItemSize.Small,
        pasteButton:{
          iconCss: "e-icons e-paste", content: "Paste",
          items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
        },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter" }
      };
    }
  };
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/itemsSize" %}

Defining items orientation

The Ribbon group orientation property allows to manage how the items are aligned either in a Row or Column. By default, the orientation is set to Column, in which the items are arranged vertically.

APP.VUE

```
<template>
  <ejs-ribbon id="ribbon">
    <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
            <e-ribbon-groups>
                <e-ribbon-group orientation="Column">
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="SplitButton"
:splitButtonSettings="pasteButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="Button"
:buttonSettings="cutButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="copyButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="formatButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                    </e-ribbon-collections>
                </e-ribbon-group>
                <e-ribbon-group orientation="Row">
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="ComboBox"
:comboBoxSettings="fontStyleSettings">
                                </e-ribbon-item>
```

```
                                    <e-ribbon-item type="ComboBox"
:comboBoxSettings="fontSizeSettings">
                                    </e-ribbon-item>
                                </e-ribbon-items>
                            </e-ribbon-collection>
                            <e-ribbon-collection>
                                <e-ribbon-items>
                                    <e-ribbon-item type="Button"
:buttonSettings="boldButton">
                                    </e-ribbon-item>
                                    <e-ribbon-item type="Button"
:buttonSettings="italicButton">
                                    </e-ribbon-item>
                                    <e-ribbon-item type="Button"
:buttonSettings="underlineButton">
                                    </e-ribbon-item>
                                </e-ribbon-items>
                            </e-ribbon-collection>
                        </e-ribbon-collections>
                    </e-ribbon-group>
                </e-ribbon-groups>
            </e-ribbon-tab>
        </e-ribbon-tabs>
    </ejs-ribbon>
</template>
<script>
    import Vue from "vue";
    import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
    Vue.use(RibbonPlugin);
    export default {
        data: function () {
            return {
                pasteButton:  {
                    iconCss: "e-icons e-paste",
                    content: "Paste",
                    items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
                },
                cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
                copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
                formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter"},
                boldButton:  { iconCss: "e-icons e-bold", content: "Bold", isToggle:
true },
                italicButton:  { iconCss: "e-icons e-italic", content: "Italic",
isToggle: true },
                underlineButton:  { iconCss: "e-icons e-underline", content:
"Underline", isToggle: true },
                fontStyleSettings:  {
                    dataSource: ["Algerian", "Arial", "Calibri", "Cambria", "Cambria
Math", "Courier New", "Candara", "Georgia", "Impact", "Segoe Print", "Segoe
Script", "Segoe UI", "Symbol", "Times New Roman", "Verdana", "Windings"],
                    index: 3
                },
                fontSizeSettings:  {
                    dataSource: ["8", "9", "10", "11", "12", "14", "16", "18", "20",
"22", "24", "26", "28", "36", "48", "72", "96"],
```

```
            index: 3
          },
        };
      }
    };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/orientation" %}

When the orientation is set to Row a group may have a maximum of three collections each of which may contain any number of items. When the orientation is set to Column a group may have any number of collections, each of which may contain one large-sized item or three medium/small-sized items. If two large-sized items are specified, it automatically converts into two medium/small-sized items.

Defining group header

You can use the header property to set the name for each group header.

APP.VUE

```
<template>
  <ejs-ribbon id="ribbon">
    <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
            <e-ribbon-groups>
                <e-ribbon-group header="Clipboard">
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="SplitButton"
:splitButtonSettings="pasteButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                    </e-ribbon-collections>
                </e-ribbon-group>
                <e-ribbon-group header="Font">
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="Button"
:buttonSettings="boldButton">
                                </e-ribbon-item>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                                    <e-ribbon-item type="Button"
:buttonSettings="italicButton">
                                    </e-ribbon-item>
                                    <e-ribbon-item type="Button"
:buttonSettings="underlineButton">
                                    </e-ribbon-item>
                                </e-ribbon-items>
                            </e-ribbon-collection>
                        </e-ribbon-collections>
                    </e-ribbon-group>
                </e-ribbon-groups>
            </e-ribbon-tab>
        </e-ribbon-tabs>
    </ejs-ribbon>
</template>
<script>
    import Vue from "vue";
    import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
    Vue.use(RibbonPlugin);
    export default {
        data: function () {
            return {
                pasteButton:  {
                    iconCss: "e-icons e-paste",
                    content: "Paste",
                    items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
                },
                boldButton:  { iconCss: "e-icons e-bold", content: "Bold", isToggle:
true },
                italicButton:  { iconCss: "e-icons e-italic", content: "Italic",
isToggle: true },
                underlineButton:  { iconCss: "e-icons e-underline", content:
"Underline", isToggle: true },
            };
        }
    };
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/groupHeader" %}

Defining group icon

You can use the groupIconCss property to customize the icons in the group overflow button. When the ribbon size is adjusted, the group popup will appear.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <ejs-ribbon id="ribbon">
    <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
            <e-ribbon-groups>
                <e-ribbon-group header="Clipboard" groupIconCss="e-icons e-
paste">
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="SplitButton"
:splitButtonSettings="pasteButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="Button"
:buttonSettings="cutButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="copyButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="formatButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                    </e-ribbon-collections>
                </e-ribbon-group>
                <e-ribbon-group header="Font" groupIconCss="e-icons e-bold">
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="ComboBox"
:comboBoxSettings="fontStyleSettings">
                                </e-ribbon-item>
                                <e-ribbon-item type="ComboBox"
:comboBoxSettings="fontSizeSettings">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="Button"
:buttonSettings="boldButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="italicButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="underlineButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    </e-ribbon-collections>
                </e-ribbon-group>
            </e-ribbon-groups>
        </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        pasteButton:  {
          iconCss: "e-icons e-paste",
          content: "Paste",
          items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
        },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter"},
        boldButton:  { iconCss: "e-icons e-bold", content: "Bold", isToggle:
true },
        italicButton:  { iconCss: "e-icons e-italic", content: "Italic",
isToggle: true },
        underlineButton:  { iconCss: "e-icons e-underline", content:
"Underline", isToggle: true },
        fontStyleSettings:  {
          dataSource: ["Algerian", "Arial", "Calibri", "Cambria", "Cambria
Math", "Courier New", "Candara", "Georgia", "Impact", "Segoe Print", "Segoe
Script", "Segoe UI", "Symbol", "Times New Roman", "Verdana", "Windings"],
          index: 3
        },
        fontSizeSettings:  {
          dataSource: ["8", "9", "10", "11", "12", "14", "16", "18", "20",
"22", "24", "26", "28", "36", "48", "72", "96"],
          index: 3
        },
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/ribbon/groupIcon" %}

Enabling group launcher icon

You can use the showLauncherIcon property to enable or disable the launcher icon for each group. By default, the property is set to false.

APP.VUE

```
<template>
  <ejs-ribbon id="ribbon">
    <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
            <e-ribbon-groups>
                <e-ribbon-group header="Clipboard" groupIconCss="e-icons e-
paste" showLauncherIcon=true>
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="Button"
:buttonSettings="cutButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="copyButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="formatButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                    </e-ribbon-collections>
                </e-ribbon-group>
            </e-ribbon-groups>
        </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter"},
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/launcherIcon" %}

Customize launcher icon

You can use the launcherIconCss property to customize the launcher icon by applying the custom styles.

APP.VUE

```
<template>
  <ejs-ribbon id="ribbon" launcherIconCss="e-icons e-description">
    <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
            <e-ribbon-groups>
                <e-ribbon-group header="Clipboard" groupIconCss="e-icons e-
paste" showLauncherIcon=true>
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="Button"
:buttonSettings="cutButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="copyButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="formatButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                    </e-ribbon-collections>
                </e-ribbon-group>
            </e-ribbon-groups>
        </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter"},
      };
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/customLauncherIcon" %}

Defining group collapsible state

You can use the isCollapsible property to determine whether the group is collapsed or not during resize. By default, the property is set to true. To prevent the group from being collapsed, set the property to false.

APP.VUE

```
<template>
  <ejs-ribbon id="ribbon">
    <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
            <e-ribbon-groups>
                <e-ribbon-group header="Clipboard" groupIconCss="e-icons e-
paste">
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="SplitButton"
:splitButtonSettings="pasteButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="Button"
:buttonSettings="cutButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="copyButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="formatButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                    </e-ribbon-collections>
                </e-ribbon-group>
                <e-ribbon-group header="Font" :isCollapsible=false>
```

```
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="ComboBox"
:comboBoxSettings="fontStyleSettings">
                                </e-ribbon-item>
                                <e-ribbon-item type="ComboBox"
:comboBoxSettings="fontSizeSettings">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="Button"
:buttonSettings="boldButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="italicButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:buttonSettings="underlineButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                    </e-ribbon-collections>
                </e-ribbon-group>
            </e-ribbon-groups>
        </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        pasteButton:  {
          iconCss: "e-icons e-paste",
          content: "Paste",
          items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
        },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter"},
        boldButton:  { iconCss: "e-icons e-bold", content: "Bold", isToggle:
true },
        italicButton:  { iconCss: "e-icons e-italic", content: "Italic",
isToggle: true },
        underlineButton:  { iconCss: "e-icons e-underline", content:
"Underline", isToggle: true },
        fontStyleSettings:  {
```

```
          dataSource: ["Algerian", "Arial", "Calibri", "Cambria", "Cambria
Math", "Courier New", "Candara", "Georgia", "Impact", "Segoe Print", "Segoe
Script", "Segoe UI", "Symbol", "Times New Roman", "Verdana", "Windings"],
          index: 3
        },
        fontSizeSettings:  {
          dataSource: ["8", "9", "10", "11", "12", "14", "16", "18", "20",
"22", "24", "26", "28", "36", "48", "72", "96"],
          index: 3
        },
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/collapsibleState" %}

Defining priority order for group collapse or expand

You can use the priority property to set the priority order for each group which should be collapsed or expanded on resizing. When collapsing, higher priority values are fetched first. When expanding, lower priority values are fetched first.

APP.VUE

```
<template>
  <ejs-ribbon id="ribbon">
    <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
            <e-ribbon-groups>
                <e-ribbon-group header="Clipboard" groupIconCss="e-icons e-
paste" priority=2>
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="SplitButton"
:splitButtonSettings="pasteButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="Button"
:buttonSettings="cutButton">
                                </e-ribbon-item>
```

```
                                    <e-ribbon-item type="Button"
:buttonSettings="copyButton">
                                    </e-ribbon-item>
                                    <e-ribbon-item type="Button"
:buttonSettings="formatButton">
                                    </e-ribbon-item>
                                </e-ribbon-items>
                            </e-ribbon-collection>
                        </e-ribbon-collections>
                    </e-ribbon-group>
                    <e-ribbon-group header="Font" groupIconCss="e-icons e-bold"
priority=0>
                        <e-ribbon-collections>
                            <e-ribbon-collection>
                                <e-ribbon-items>
                                    <e-ribbon-item type="ComboBox"
:comboBoxSettings="fontStyleSettings">
                                    </e-ribbon-item>
                                    <e-ribbon-item type="ComboBox"
:comboBoxSettings="fontSizeSettings">
                                    </e-ribbon-item>
                                </e-ribbon-items>
                            </e-ribbon-collection>
                            <e-ribbon-collection>
                                <e-ribbon-items>
                                    <e-ribbon-item type="Button"
:buttonSettings="boldButton">
                                    </e-ribbon-item>
                                    <e-ribbon-item type="Button"
:buttonSettings="italicButton">
                                    </e-ribbon-item>
                                    <e-ribbon-item type="Button"
:buttonSettings="underlineButton">
                                    </e-ribbon-item>
                                </e-ribbon-items>
                            </e-ribbon-collection>
                        </e-ribbon-collections>
                    </e-ribbon-group>
                    <e-ribbon-group header="Editing" groupIconCss="e-icons e-
edit" priority=1>
                        <e-ribbon-collections>
                          <e-ribbon-collection>
                            <e-ribbon-items>
                              <e-ribbon-item type="SplitButton"
:splitButtonSettings="searchOptions" >
                              </e-ribbon-item>
                              <e-ribbon-item type="Button"
:buttonSettings="replaceButton">
                              </e-ribbon-item>
                              <e-ribbon-item type="SplitButton"
:splitButtonSettings="selectOptions" >
                              </e-ribbon-item>
                            </e-ribbon-items>
                          </e-ribbon-collection>
                        </e-ribbon-collections>
                      </e-ribbon-group>
                </e-ribbon-groups>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        pasteButton:  {
          iconCss: "e-icons e-paste",
          content: "Paste",
          items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
        },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter"},
        boldButton:  { iconCss: "e-icons e-bold", content: "Bold", isToggle:
true },
        italicButton:  { iconCss: "e-icons e-italic", content: "Italic",
isToggle: true },
        underlineButton:  { iconCss: "e-icons e-underline", content:
"Underline", isToggle: true },
        fontStyleSettings:  {
          dataSource: ["Algerian", "Arial", "Calibri", "Cambria", "Cambria
Math", "Courier New", "Candara", "Georgia", "Impact", "Segoe Print", "Segoe
Script", "Segoe UI", "Symbol", "Times New Roman", "Verdana", "Windings"],
          index: 3
        },
        fontSizeSettings:  {
          dataSource: ["8", "9", "10", "11", "12", "14", "16", "18", "20",
"22", "24", "26", "28", "36", "48", "72", "96"],
          index: 3
        },
        searchOptions:{
        iconCss: 'e-icons e-search', content: 'Find',
        items: [
          { text: 'Find', iconCss: 'e-icons e-search' },
          { text: 'Advanced find', iconCss: 'e-icons e-search' },
          { text: 'Go to', iconCss: 'e-icons e-arrow-right' }]
        },
        replaceButton: { iconCss: "e-icons e-replace", content: "Replace" },
        selectOptions: {
          iconCss: "e-icons e-mouse-pointer", content: "Select",
          items: [{ text: "Select All" }, { text: "Select Objects" }],
        },
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/priority" %}

Simplified layout

In simplified layout, the Ribbon component organizes the items and groups into a single row by setting the activeLayout property to Simplified.

APP.VUE

```
<template>
  <ejs-ribbon activeLayout="Simplified">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Clipboard">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="SplitButton"
:allowedSizes="largeSize" :splitButtonSettings="pasteSettigs" >
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
      <e-ribbon-tab header="Insert">
        <e-ribbon-groups>
          <e-ribbon-group header="Tables">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button" :buttonSettings="copyButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button"
:buttonSettings="formatButton">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonItemSize, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        largeSize: RibbonItemSize.Large,
        pasteSettigs:{
          iconCss: "e-icons e-paste", content: "Paste",
          items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
        },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter" }
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/simplified" %}

Enabling group overflow popup

You can use the enableGroupOverflow property to add a separate popup for the overflow items in the group while resizing. The overflow items will appear in the common overflow popup, located at the right end of the tab content if it is set to false.

APP.VUE

```
<template>
  <ejs-ribbon id="ribbon">
    <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
            <e-ribbon-groups>
                <e-ribbon-group header="Clipboard" groupIconCss="e-icons e-
paste">
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
```

```
                                    <e-ribbon-item type="SplitButton"
:splitButtonSettings="pasteButton">
                                    </e-ribbon-item>
                                </e-ribbon-items>
                            </e-ribbon-collection>
                            <e-ribbon-collection>
                                <e-ribbon-items>
                                    <e-ribbon-item type="Button"
:buttonSettings="cutButton">
                                    </e-ribbon-item>
                                    <e-ribbon-item type="Button"
:buttonSettings="copyButton">
                                    </e-ribbon-item>
                                    <e-ribbon-item type="Button"
:buttonSettings="formatButton">
                                    </e-ribbon-item>
                                </e-ribbon-items>
                            </e-ribbon-collection>
                        </e-ribbon-collections>
                    </e-ribbon-group>
                    <e-ribbon-group header="Font" orientation=Row
groupIconCss="e-icons e-bold" enableGroupOverflow=true>
                        <e-ribbon-collections>
                            <e-ribbon-collection>
                                <e-ribbon-items>
                                    <e-ribbon-item type="ComboBox"
:comboBoxSettings="fontStyleSettings">
                                    </e-ribbon-item>
                                    <e-ribbon-item type="ComboBox"
:comboBoxSettings="fontSizeSettings">
                                    </e-ribbon-item>
                                </e-ribbon-items>
                            </e-ribbon-collection>
                            <e-ribbon-collection>
                                <e-ribbon-items>
                                    <e-ribbon-item type="Button"
:buttonSettings="boldButton">
                                    </e-ribbon-item>
                                    <e-ribbon-item type="Button"
:buttonSettings="italicButton">
                                    </e-ribbon-item>
                                    <e-ribbon-item type="Button"
:buttonSettings="underlineButton">
                                    </e-ribbon-item>
                                </e-ribbon-items>
                            </e-ribbon-collection>
                        </e-ribbon-collections>
                    </e-ribbon-group>
                    <e-ribbon-group header="Editing" groupIconCss="e-icons e-
edit">
                      <e-ribbon-collections>
                        <e-ribbon-collection>
                          <e-ribbon-items>
                            <e-ribbon-item type="SplitButton"
:splitButtonSettings="searchOptions" >
                            </e-ribbon-item>
```

```
                        <e-ribbon-item type="Button"
:buttonSettings="replaceButton">
                        </e-ribbon-item>
                        <e-ribbon-item type="SplitButton"
:splitButtonSettings="selectOptions" >
                        </e-ribbon-item>
                      </e-ribbon-items>
                    </e-ribbon-collection>
                  </e-ribbon-collections>
                </e-ribbon-group>
              </e-ribbon-groups>
          </e-ribbon-tab>
      </e-ribbon-tabs>
    </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        pasteButton:  {
          iconCss: "e-icons e-paste",
          content: "Paste",
          items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
        },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter"},
        boldButton:  { iconCss: "e-icons e-bold", content: "Bold", isToggle:
true },
        italicButton:  { iconCss: "e-icons e-italic", content: "Italic",
isToggle: true },
        underlineButton:  { iconCss: "e-icons e-underline", content:
"Underline", isToggle: true },
        fontStyleSettings:  {
          dataSource: ["Algerian", "Arial", "Calibri", "Cambria", "Cambria
Math", "Courier New", "Candara", "Georgia", "Impact", "Segoe Print", "Segoe
Script", "Segoe UI", "Symbol", "Times New Roman", "Verdana", "Windings"],
          index: 3
        },
        fontSizeSettings:  {
          dataSource: ["8", "9", "10", "11", "12", "14", "16", "18", "20",
"22", "24", "26", "28", "36", "48", "72", "96"],
          index: 3
        },
        searchOptions:{
        iconCss: 'e-icons e-search', content: 'Find',
        items: [
          { text: 'Find', iconCss: 'e-icons e-search' },
          { text: 'Advanced find', iconCss: 'e-icons e-search' },
          { text: 'Go to', iconCss: 'e-icons e-arrow-right' }]
        },
        replaceButton: { iconCss: "e-icons e-replace", content: "Replace" },
```

```
        selectOptions: {
          iconCss: "e-icons e-mouse-pointer", content: "Select",
          items: [{ text: "Select All" }, { text: "Select Objects" }],
        },
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/groupOverflow" %}

Minimized State

You can hide the Ribbon contents and display only the tab headers by double-clicking on the tab header. In minimized state, the Ribbon component expands to its normal state when click on the tab header.

You can use the isMinimized property to change the Ribbon component to minimized state. By default, the value is false.

APP.VUE

```
<template>
  <ejs-ribbon isMinimized="true">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Clipboard">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="SplitButton"
:allowedSizes="largeSize" :splitButtonSettings="pasteSettigs" >
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
      <e-ribbon-tab header="Insert">
        <e-ribbon-groups>
          <e-ribbon-group header="Tables">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <e-ribbon-item type="Button" :buttonSettings="cutButton">
                    </e-ribbon-item>
                    <e-ribbon-item type="Button" :buttonSettings="copyButton">
                    </e-ribbon-item>
                    <e-ribbon-item type="Button"
:buttonSettings="formatButton">
                    </e-ribbon-item>
                  </e-ribbon-items>
                </e-ribbon-collection>
              </e-ribbon-collections>
            </e-ribbon-group>
          </e-ribbon-groups>
        </e-ribbon-tab>
      </e-ribbon-tabs>
    </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonItemSize, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        largeSize: RibbonItemSize.Large,
        pasteSettigs:{
          iconCss: "e-icons e-paste", content: "Paste",
          items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
        },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter" }
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/isMinimized" %}

File Menu

The Ribbon component provides a built-in file menu that allows you to add menu items for performing specific actions. The file menu can be enabled by setting the fileMenu property.

Copyright © 2001 -2024 Syncfusion Inc.

Visibility

You can show the file menu by setting the visible property to true. By default, the file menu is hidden.

APP.VUE

```
<template>
  <ejs-ribbon :fileMenu="fileSettings">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Font">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="ColorPicker"
:colorPickerSettings="colorPicker" >
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button" :buttonSettings="copyButton">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonFileMenu, RibbonPlugin, RibbonColorPicker } from
"@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
  provide: {
    ribbon: [RibbonFileMenu, RibbonColorPicker]
  },
  data: function () {
    return {
      fileSettings: {
        visible: true,
        menuItems: [
          { text: "New", iconCss: "e-icons e-file-new", id: "new" }
        ]
      },
      cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
      copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
      colorPicker: {value: "#123456" },
    };
  }
  };
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/modules" %}

Adding menu items

The menu items can be added to the file menu using the menuItems property as an array of MenuItemModel.

APP.VUE

```
<template>
  <ejs-ribbon :fileMenu="fileSettings">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Font">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button" :buttonSettings="copyButton">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonFileMenu, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
  provide: {
    ribbon: [RibbonFileMenu]
  },
  data: function () {
    return {
      fileSettings: {
        visible: true,
```

```
        menuItems: [
          { text: "New", iconCss: "e-icons e-file-new", id: "new" },
          { text: "Open", iconCss: "e-icons e-folder-open", id: "open" },
          { text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
          { text: "Save as", iconCss: "e-icons e-save", id: "save" }
        ]
      },
      cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
      copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
    };
  }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/menuItems" %}

Open submenu on click

You can open the submenu on menu item click, by setting the showItemOnClick property to true. By default, the submenu will open on mouse hover.

APP.VUE

```
<template>
  <ejs-ribbon :fileMenu="fileSettings">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Font">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button" :buttonSettings="copyButton">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
  import Vue from "vue";
  import { RibbonFileMenu, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
  provide: {
    ribbon: [RibbonFileMenu]
  },
  data: function () {
    return {
      fileSettings: {
        showItemOnClick: true,
        visible: true,
        menuItems: [
          { text: "New", iconCss: "e-icons e-file-new", id: "new" },
          { text: "Open", iconCss: "e-icons e-folder-open", id: "open" },
          { text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
          {
            text: 'Save as',
            iconCss:'e-icons e-save',
            id: 'save',
            items: [
                { text: 'Microsoft Word (.docx)' },
                { text: 'Microsoft Word 97-2003(.doc)' },
                { text: 'Download as PDF' }
            ]
          }
        ]
      },
      cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
      copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
    };
  }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/subMenuOnClick" %}

Custom header text

You can define the file menu header text content by using the text property.

APP.VUE

```
<template>
```

```
  <ejs-ribbon :fileMenu="fileSettings">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Font">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button" :buttonSettings="copyButton">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonFileMenu, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
  provide: {
    ribbon: [RibbonFileMenu]
  },
  data: function () {
    return {
      fileSettings: {
        text: 'App',
        visible: true,
        menuItems: [
          { text: "New", iconCss: "e-icons e-file-new", id: "new" },
          { text: "Open", iconCss: "e-icons e-folder-open", id: "open" },
          { text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
          { text: "Save as", iconCss: "e-icons e-save", id: "save" }
        ]
      },
      cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
      copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
    };
  }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/headerText" %}

Ribbon Backstage

The Ribbon supports backstage view which is an addition to the traditional file menu. It displays information like application settings, user details, etc. The backstage view can be enabled by setting the backStageMenu property.

The backstage view options are displayed on the left, while the content of each option is shown on the right.

Adding backstage items

The menu items can be added to the backstage view by using the items property. You can show the backstage view by setting the visible property to true. By default, the backstage view is hidden.

APP.VUE

```
<template>
  <ejs-ribbon ref="backstageRibbon"  :backStageMenu="backstageSettings">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Clipboard">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button" :buttonSettings="copyButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button"
:buttonSettings="pasteButton">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonBackstage } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
  provide: {
      ribbon: [RibbonBackstage]
  },
  data: function () {
      return {
        backstageSettings: {
          visible: true,
```

```
          items: [
            { id: 'home', text: 'Home', iconCss: 'e-icons e-home', content:
this.homeContentTemplate() },
            { id: 'new', text: 'New', iconCss: 'e-icons e-file-new',
content: this.newContentTemplate() },
            { id: 'open', text: 'Open', iconCss: 'e-icons e-folder-open',
content: this.openContentTemplate() }
          ],
          backButton: {
            text: 'Close',
          }
      },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton: { iconCss: "e-icons e-copy", content: "Copy"},
        pasteButton: { iconCss: "e-icons e-paste", content: "Paste"},
      };
    },
    methods: {
      homeContentTemplate() {
        return `
        <div id='home-wrapper' style="padding: 20px;">
          <div id='new-section' class='new-wrapper'>
              <div class='section-title'> New </div>
              <div class='category_container'>
                  <div class='doc_category_image'></div>
                  <span class='doc_category_text'> New document </span>
              </div>
          </div>
          <div id='block-wrapper'><div class='section-title'> Recent </div>
              <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                  <table>
                      <tbody>
                          <tr>
                              <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                              <td>
                                  <span style="display: block; font-size:
14px"> Ribbon.docx </span>
                                  <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
                              </td>
                          </tr>
                      </tbody>
                  </table>
              </div>
          </div>
      </div>
        `;
      },
      newContentTemplate() {
        return `
        <div id='new-content' style="padding: 20px;">
          <div id='new-section' class='new-wrapper'>
              <div class='section-title'> New </div>
              <div class='category_container'>
                  <div class='doc_category_image'></div>
```

```
                    <span class='doc_category_text'> New document </span>
                </div>
            </div>
        </div>
          `;
        },
        openContentTemplate() {
          return `
          <div style="padding: 20px;">
            <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                <table>
                    <tbody>
                        <tr>
                            <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                            <td>
                                <span style="display: block; font-size: 14px">
Open in Desktop App </span>
                                <span style="font-size: 12px"> Use the full
functionality of Ribbon </span>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
            <div class="section-content" style="padding: 12px 0px cursor:
pointer">
                <table>
                    <tbody>
                        <tr>
                            <td> <span class="doc_icon e-icons e-protect-
sheet"></span> </td>
                            <td>
                                <span style="display: block; font-size: 14px">
Protect Document </span>
                                <span style="font-size: 12px">To prevent
accidental changes,
                                    this document has been set to open as
view-only.</span>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
          `;
        },
    }
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
.e-ribbon-backstage-content{
    width: 550px;
    height: 350px;
}
.section-title {
    font-size: 22px;
}
.new-docs {
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
}
.category_container {
    width: 150px;
    padding: 15px;
    text-align: center;
    cursor: pointer;
}
.doc_category_image {
    width: 80px;
    height: 100px;
    background-color: #fff;
    border: 1px solid rgb(125, 124, 124);
    text-align: center;
    overflow: hidden;
    margin: 0px auto 10px;
}
.doc_category_text {
    font-size: 16px;
}
.section-content {
    padding: 12px 0px;
    cursor: pointer;
}
.doc_icon {
    font-size: 16px;
    padding: 0px 10px;
}
.category_container:hover, .section-content:hover {
    background-color: #dfdfdf;
    border-radius: 5px;
    transition: all 0.3s;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/backStageItem" %}

Copyright © 2001 -2024 Syncfusion Inc.

Adding footer items

You can use the isFooter property in the items collection to add the backstage view footer items. By default, the value is false.

APP.VUE

```
<template>
  <ejs-ribbon ref="backstageRibbon"  :backStageMenu="backstageSettings">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Clipboard">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button" :buttonSettings="copyButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button"
:buttonSettings="pasteButton">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonBackstage } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
  provide: {
      ribbon: [RibbonBackstage]
  },
  data: function () {
      return {
        backstageSettings: {
          visible: true,
          items: [
            { id: 'home', text: 'Home', iconCss: 'e-icons e-home', content:
this.homeContentTemplate() },
            { id: 'new', text: 'New', iconCss: 'e-icons e-file-new',
content: this.newContentTemplate() },
            { id: 'open', text: 'Open', iconCss: 'e-icons e-folder-open',
content: this.openContentTemplate() },
            { separator: true, isFooter: true },
            { id: 'account', text: 'Account', isFooter: true, content:
this.accountContent() }
          ],
          backButton: {
            text: 'Close',
          }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton: { iconCss: "e-icons e-copy", content: "Copy"},
        pasteButton: { iconCss: "e-icons e-paste", content: "Paste"},
      };
    },
    methods: {
      homeContentTemplate() {
        return `
        <div id='home-wrapper' style="padding: 20px;">
          <div id='new-section' class='new-wrapper'>
              <div class='section-title'> New </div>
              <div class='category_container'>
                  <div class='doc_category_image'></div>
                  <span class='doc_category_text'> New document </span>
              </div>
          </div>
          <div id='block-wrapper'><div class='section-title'> Recent </div>
              <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                  <table>
                      <tbody>
                          <tr>
                              <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                              <td>
                                  <span style="display: block; font-size:
14px"> Ribbon.docx </span>
                                  <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
                              </td>
                          </tr>
                      </tbody>
                  </table>
              </div>
          </div>
      </div>
        `;
      },
      newContentTemplate() {
        return `
        <div id='new-content' style="padding: 20px;">
          <div id='new-section' class='new-wrapper'>
              <div class='section-title'> New </div>
              <div class='category_container'>
                  <div class='doc_category_image'></div>
                  <span class='doc_category_text'> New document </span>
              </div>
          </div>
      </div>
        `;
      },
      openContentTemplate() {
        return `
        <div style="padding: 20px;">
          <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
```

```
                <table>
                    <tbody>
                        <tr>
                            <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                            <td>
                                <span style="display: block; font-size: 14px">
Open in Desktop App </span>
                                <span style="font-size: 12px"> Use the full
functionality of Ribbon </span>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
            <div class="section-content" style="padding: 12px 0px cursor:
pointer">
                <table>
                    <tbody>
                        <tr>
                            <td> <span class="doc_icon e-icons e-protect-
sheet"></span> </td>
                            <td>
                                <span style="display: block; font-size: 14px">
Protect Document </span>
                                <span style="font-size: 12px">To prevent
accidental changes,
                                    this document has been set to open as
view-only.</span>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
          `;
        },
        accountContent () {
          return `
          <div style="padding: 20px;">
          <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
              <table>
                  <tbody>
                      <tr>
                          <td> <span class="doc_icon e-icons e-people"></span>
</td>
                          <td>
                              <span style="display: block; font-size:
14px">Account type</span>
                              <span style="font-size:
12px">Administrator</span>
                          </td>
                      </tr>
                  </tbody>
              </table>
          </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <div class="section-content" style="padding: 12px 0px cursor:
pointer">
            <table>
                <tbody>
                    <tr>
                        <td> <span class="doc_icon e-icons e-
password"></span> </td>
                        <td>
                            <span style="display: block; font-size:
14px">Password protected</span>
                            <span style="font-size: 12px">Yes</span>
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
        `;
      }
  }
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
.e-ribbon-backstage-content{
    width: 550px;
    height: 350px;
}
.section-title {
    font-size: 22px;
}
.new-docs {
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
}
.category_container {
    width: 150px;
    padding: 15px;
    text-align: center;
    cursor: pointer;
}
.doc_category_image {
    width: 80px;
    height: 100px;
    background-color: #fff;
    border: 1px solid rgb(125, 124, 124);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    text-align: center;
    overflow: hidden;
    margin: 0px auto 10px;
}
.doc_category_text {
    font-size: 16px;
}
.section-content {
    padding: 12px 0px;
    cursor: pointer;
}
.doc_icon {
    font-size: 16px;
    padding: 0px 10px;
}
.category_container:hover, .section-content:hover {
    background-color: #dfdfdf;
    border-radius: 5px;
    transition: all 0.3s;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/backstageFooter-item" %}

Adding separator

The separators are horizontal lines used to separate the backstage view items. You can use the separator property to split the menu items.

APP.VUE

```
<template>
  <ejs-ribbon ref="backstageRibbon"  :backStageMenu="backstageSettings">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Clipboard">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button" :buttonSettings="copyButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button"
:buttonSettings="pasteButton">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
```

```
import Vue from "vue";
import { RibbonPlugin, RibbonBackstage } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
  provide: {
      ribbon: [RibbonBackstage]
  },
  data: function () {
      return {
        backstageSettings: {
          visible: true,
          items: [
            { id: 'home', text: 'Home', iconCss: 'e-icons e-home', content:
this.homeContentTemplate() },
            { id: 'new', text: 'New', iconCss: 'e-icons e-file-new',
content: this.newContentTemplate() },
            { id: 'open', text: 'Open', iconCss: 'e-icons e-folder-open',
content: this.openContentTemplate() },
            { separator: true },
            { text: 'Print', content: this.printContent() }
          ],
          backButton: {
            text: 'Close',
          }
      },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton: { iconCss: "e-icons e-copy", content: "Copy"},
        pasteButton: { iconCss: "e-icons e-paste", content: "Paste"},
      };
    },
    methods: {
      homeContentTemplate() {
        return `
        <div id='home-wrapper' style="padding: 20px;">
          <div id='new-section' class='new-wrapper'>
              <div class='section-title'> New </div>
              <div class='category_container'>
                  <div class='doc_category_image'></div>
                  <span class='doc_category_text'> New document </span>
              </div>
          </div>
          <div id='block-wrapper'><div class='section-title'> Recent </div>
              <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                  <table>
                      <tbody>
                          <tr>
                              <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                              <td>
                                  <span style="display: block; font-size:
14px"> Ribbon.docx </span>
                                  <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
                              </td>
                          </tr>
                      </tbody>
```

```
                    </table>
                </div>
            </div>
        </div>
          `;
        },
        newContentTemplate() {
          return `
          <div id='new-content' style="padding: 20px;">
            <div id='new-section' class='new-wrapper'>
                <div class='section-title'> New </div>
                <div class='category_container'>
                    <div class='doc_category_image'></div>
                    <span class='doc_category_text'> New document </span>
                </div>
            </div>
        </div>
          `;
        },
        openContentTemplate() {
          return `
          <div style="padding: 20px;">
            <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                <table>
                    <tbody>
                        <tr>
                            <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                            <td>
                                <span style="display: block; font-size: 14px">
Open in Desktop App </span>
                                <span style="font-size: 12px"> Use the full
functionality of Ribbon </span>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
            <div class="section-content" style="padding: 12px 0px cursor:
pointer">
                <table>
                    <tbody>
                        <tr>
                            <td> <span class="doc_icon e-icons e-protect-
sheet"></span> </td>
                            <td>
                                <span style="display: block; font-size: 14px">
Protect Document </span>
                                <span style="font-size: 12px">To prevent
accidental changes,
                                    this document has been set to open as
view-only.</span>
                            </td>
                        </tr>
                    </tbody>
                </table>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            </div>
        </div>
          `;
        },
        printContent () {
          return `
          <div style="min-width: 300px; padding: 20px;">
            <h2>Print this document</h2>
            <button id="togglebtn" class="e-control e-btn e-lib e-flat e-
primary">
            <span class="e-btn-icon e-btn-sb-icons e-icons e-print e-icon-
left">
            </span>Print</button>
          </div>
          `;
        }
    }
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
.e-ribbon-backstage-content{
    width: 350px;
    height: 530px;
}
.section-title {
    font-size: 22px;
}
.new-docs {
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
}
.category_container {
    width: 150px;
    padding: 15px;
    text-align: center;
    cursor: pointer;
}
.doc_category_image {
    width: 80px;
    height: 100px;
    background-color: #fff;
    border: 1px solid rgb(125, 124, 124);
    text-align: center;
    overflow: hidden;
    margin: 0px auto 10px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
.doc_category_text {
    font-size: 16px;
}
.section-content {
    padding: 12px 0px;
    cursor: pointer;
}
.doc_icon {
    font-size: 16px;
    padding: 0px 10px;
}
.category_container:hover, .section-content:hover {
    background-color: #dfdfdf;
    border-radius: 5px;
    transition: all 0.3s;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/backstageSeparator" %}

Back button

You can use the backButton property to customize the text and icon of the close button using the text and iconCss property. You can show the back button by setting the visible property to true.

APP.VUE

```
<template>
  <ejs-ribbon ref="backstageRibbon"  :backStageMenu="backstageSettings">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Clipboard">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button" :buttonSettings="copyButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button"
:buttonSettings="pasteButton">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonBackstage } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  provide: {
      ribbon: [RibbonBackstage]
  },
  data: function () {
      return {
        backstageSettings: {
          visible: true,
          items: [
            { id: 'home', text: 'Home', iconCss: 'e-icons e-home', content:
this.homeContentTemplate() },
          ],
          backButton: {
            text: 'Close',
          }
      },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton: { iconCss: "e-icons e-copy", content: "Copy"},
        pasteButton: { iconCss: "e-icons e-paste", content: "Paste"},
      };
    },
    methods: {
      homeContentTemplate() {
        return `
        <div id='home-wrapper' style="padding: 20px;">
          <div id='new-section' class='new-wrapper'>
              <div class='section-title'> New </div>
              <div class='category_container'>
                  <div class='doc_category_image'></div>
                  <span class='doc_category_text'> New document </span>
              </div>
          </div>
          <div id='block-wrapper'><div class='section-title'> Recent </div>
              <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                  <table>
                      <tbody>
                          <tr>
                              <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                              <td>
                                  <span style="display: block; font-size:
14px"> Ribbon.docx </span>
                                  <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
                              </td>
                          </tr>
                      </tbody>
                  </table>
              </div>
          </div>
      </div>
        `;
      },
  }
};
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
.e-ribbon-backstage-content{
    width: 550px;
    height: 350px;
}
.section-title {
    font-size: 22px;
}
.new-docs {
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
}
.category_container {
    width: 150px;
    padding: 15px;
    text-align: center;
    cursor: pointer;
}
.doc_category_image {
    width: 80px;
    height: 100px;
    background-color: #fff;
    border: 1px solid rgb(125, 124, 124);
    text-align: center;
    overflow: hidden;
    margin: 0px auto 10px;
}
.doc_category_text {
    font-size: 16px;
}
.section-content {
    padding: 12px 0px;
    cursor: pointer;
}
.doc_icon {
    font-size: 16px;
    padding: 0px 10px;
}
.category_container:hover, .section-content:hover {
    background-color: #dfdfdf;
    border-radius: 5px;
    transition: all 0.3s;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/ribbon/backstageBackButton" %}

Backstage target

The target property specifies the element selector in which backstage will be displayed. The target element should have the position as relative, else the backstage will be positioned nearest to the relative element. By default, the backstage is positioned to ribbon element.

APP.VUE

```
<template>
  <div id="ribbon-backstage-section">
    <ejs-ribbon ref="backstageRibbon"  :backStageMenu="backstageSettings">
      <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
          <e-ribbon-groups>
            <e-ribbon-group header="Clipboard">
              <e-ribbon-collections>
                <e-ribbon-collection>
                  <e-ribbon-items>
                    <e-ribbon-item type="Button"
:buttonSettings="cutButton">
                    </e-ribbon-item>
                    <e-ribbon-item type="Button"
:buttonSettings="copyButton">
                    </e-ribbon-item>
                    <e-ribbon-item type="Button"
:buttonSettings="pasteButton">
                    </e-ribbon-item>
                  </e-ribbon-items>
                </e-ribbon-collection>
              </e-ribbon-collections>
            </e-ribbon-group>
          </e-ribbon-groups>
        </e-ribbon-tab>
      </e-ribbon-tabs>
    </ejs-ribbon>
    <div id="targetElement"></div>
  </div>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonBackstage } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
  provide: {
      ribbon: [RibbonBackstage]
  },
  data: function () {
      return {
        backstageSettings: {
          target: '#targetElement',
          visible: true,
          items: [
            { id: 'home', text: 'Home', iconCss: 'e-icons e-home', content:
this.homeContentTemplate() },
```

```
            { id: 'new', text: 'New', iconCss: 'e-icons e-file-new',
content: this.newContentTemplate() },
            { id: 'open', text: 'Open', iconCss: 'e-icons e-folder-open',
content: this.openContentTemplate() }
          ],
          backButton: {
            text: 'Close',
          }
      },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton: { iconCss: "e-icons e-copy", content: "Copy"},
        pasteButton: { iconCss: "e-icons e-paste", content: "Paste"},
      };
    },
    methods: {
      homeContentTemplate() {
        return `
        <div id='home-wrapper' style="padding: 20px;">
          <div id='new-section' class='new-wrapper'>
              <div class='section-title'> New </div>
              <div class='category_container'>
                  <div class='doc_category_image'></div>
                  <span class='doc_category_text'> New document </span>
              </div>
          </div>
          <div id='block-wrapper'><div class='section-title'> Recent </div>
              <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                  <table>
                      <tbody>
                          <tr>
                              <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                              <td>
                                  <span style="display: block; font-size:
14px"> Ribbon.docx </span>
                                  <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
                              </td>
                          </tr>
                      </tbody>
                  </table>
              </div>
          </div>
      </div>
        `;
      },
      newContentTemplate() {
        return `
        <div id='new-content' style="padding: 20px;">
          <div id='new-section' class='new-wrapper'>
              <div class='section-title'> New </div>
              <div class='category_container'>
                  <div class='doc_category_image'></div>
                  <span class='doc_category_text'> New document </span>
              </div>
          </div>
```

```
      </div>
        `;
      },
      openContentTemplate() {
        return `
        <div style="padding: 20px;">
          <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
              <table>
                  <tbody>
                      <tr>
                          <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                          <td>
                              <span style="display: block; font-size: 14px">
Open in Desktop App </span>
                              <span style="font-size: 12px"> Use the full
functionality of Ribbon </span>
                          </td>
                      </tr>
                  </tbody>
              </table>
          </div>
          <div class="section-content" style="padding: 12px 0px cursor:
pointer">
              <table>
                  <tbody>
                      <tr>
                          <td> <span class="doc_icon e-icons e-protect-
sheet"></span> </td>
                          <td>
                              <span style="display: block; font-size: 14px">
Protect Document </span>
                              <span style="font-size: 12px">To prevent
accidental changes,
                                  this document has been set to open as
view-only.</span>
                          </td>
                      </tr>
                  </tbody>
              </table>
          </div>
      </div>
        `;
      },
  }
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
.e-ribbon-backstage-content{
    width: 550px;
    height: 530px;
}
#targetElement{
    width: 300px;
    height: 300px;
}
.section-title {
    font-size: 22px;
}
.new-docs {
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
}
.category_container {
    width: 150px;
    padding: 15px;
    text-align: center;
    cursor: pointer;
}
.doc_category_image {
    width: 80px;
    height: 100px;
    background-color: #fff;
    border: 1px solid rgb(125, 124, 124);
    text-align: center;
    overflow: hidden;
    margin: 0px auto 10px;
}
.doc_category_text {
    font-size: 16px;
}
.section-content {
    padding: 12px 0px;
    cursor: pointer;
}
.doc_icon {
    font-size: 16px;
    padding: 0px 10px;
}
.category_container:hover, .section-content:hover {
    background-color: #dfdfdf;
    border-radius: 5px;
    transition: all 0.3s;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/backstageTarget" %}

Template

You can use the template property to modify the backstage view menu items and their contents.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <ejs-ribbon id="ribbon" ref="backstageRibbon"
:backStageMenu="backstageSettings">
      <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
          <e-ribbon-groups>
            <e-ribbon-group header="Clipboard">
              <e-ribbon-collections>
                <e-ribbon-collection>
                  <e-ribbon-items>
                    <e-ribbon-item type="Button"
:buttonSettings="cutButton">
                    </e-ribbon-item>
                    <e-ribbon-item type="Button"
:buttonSettings="copyButton">
                    </e-ribbon-item>
                    <e-ribbon-item type="Button"
:buttonSettings="pasteButton">
                    </e-ribbon-item>
                  </e-ribbon-items>
                </e-ribbon-collection>
              </e-ribbon-collections>
            </e-ribbon-group>
          </e-ribbon-groups>
        </e-ribbon-tab>
      </e-ribbon-tabs>
    </ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonBackstage } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
  provide: {
  ribbon: [RibbonBackstage]
},
data: function () {
  return {
    backstageSettings: {
      visible: true,
      template: this.homeContentTemplate()
    },
    cutButton: { iconCss: "e-icons e-cut", content: "Cut" },
    copyButton: { iconCss: "e-icons e-copy", content: "Copy" },
    pasteButton: { iconCss: "e-icons e-paste", content: "Paste" },
  };
},
methods: {
  displayPopup() {
    this.backstagePopup =
this.ribbonEle.querySelector('#ribbon_backstagepopup');
    if (this.backstagePopup) {
      this.backstagePopup.style.display = 'block';
    }
  },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  contentClick (id) {
      this.content = this.ribbonEle.querySelector('.content-open')
      if(this.content) { this.content.classList.replace('content-open',
'content-close'); }
      this.ribbonEle.querySelector('#' + id +'-
wrapper').classList.add('content-open');
  },
  closeContent () {
      this.ribbonEle.querySelector('#ribbon_backstagepopup').style.display =
'none'
  },
  homeContentTemplate() {
    return `
    <div id="temp-content" style="width: 550px; height: 350px; display:
flex">
      <div id="items-wrapper" style="width: 130px; height:100%; background:
#779de8;">
          <ul>
              <li id="close" @click="closeContent">
                  <span class="e-icons e-close"></span>
                  Close
              </li>
              <li id="new" @click="contentClick('new')">
                  <span class="e-icons e-file-new"></span>
                  New
              </li>
              <li id="open" @click="contentClick('open')">
                  <span class="e-icons e-folder-open"></span>
                  Open
              </li>
              <li id="save" @click="contentClick('save')">
                  <span class="e-icons e-save"></span>
                  Save
              </li>
          </ul>
      </div>
      <div id="content-wrapper">
          <div id='new-wrapper' class='content-open' style="padding: 20px;">
              <div id='new-section' class='new-wrapper'>
                  <div class='section-title'> New </div>
                  <div class='category_container'>
                      <div class='doc_category_image'></div>
                      <span class='doc_category_text'> New document </span>
                  </div>
              </div>
          </div>
          <div id="save-wrapper" class='content-close' style="padding:
20px;">
              <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                  <table>
                      <tbody>
                          <tr>
                              <td> <span class="doc_icon e-icons e-
save"></span> </td>
                              <td>
```

```
                                    <span style="display: block; font-size:
14px"> Save as </span>
                                    <span style="font-size: 12px"> Save as
copy online </span>
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </div>
                <div class="section-content" style="padding: 12px 0px cursor:
pointer">
                    <table>
                        <tbody>
                            <tr>
                                <td> <span class="doc_icon e-icons e-
rename"></span> </td>
                                <td>
                                    <span style="display: block; font-size:
14px"> Rename </span>
                                    <span style="font-size: 12px">Rename this
file. </span>
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>
            <div id="open-wrapper" class='content-close' style="padding:
20px;">
                <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                    <table>
                        <tbody>
                            <tr>
                                <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                                <td>
                                    <span style="display: block; font-size:
14px"> Ribbon.docx </span>
                                    <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </div>
                <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                    <table>
                        <tbody>
                            <tr>
                                <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                                <td>
                                    <span style="display: block; font-size:
14px"> Classic_layout.docx </span>
```

```
                                    <span style="font-size: 12px">  EJ2 >>
Components >> Navigations >> Ribbon >> layouts </span>
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </div>
                <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                    <table>
                        <tbody>
                            <tr>
                                <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                                <td>
                                    <span style="display: block; font-size:
14px"> Simplified_layout.docx </span>
                                    <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> layouts </span>
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
        `;
    },
},
mounted() {
    this.ribbonEle = document.getElementById('ribbon');
    document.querySelector('.e-ribbon-backstage').onClick =
this.displayPopup.bind(this);
},
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
.e-ribbon-backstage-content{
    width: 550px;
    height: 350px;
}
.section-title {
    font-size: 22px;
}
.new-docs {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
}
.category_container {
    width: 150px;
    padding: 15px;
    text-align: center;
    cursor: pointer;
}
.doc_category_image {
    width: 80px;
    height: 100px;
    background-color: #fff;
    border: 1px solid rgb(125, 124, 124);
    text-align: center;
    overflow: hidden;
    margin: 0px auto 10px;
}
.doc_category_text {
    font-size: 16px;
}
.section-content {
    padding: 12px 0px;
    cursor: pointer;
}
.doc_icon {
    font-size: 16px;
    padding: 0px 10px;
}
.category_container:hover, .section-content:hover {
    background-color: #dfdfdf;
    border-radius: 5px;
    transition: all 0.3s;
}
#targetElement{
    width: 500px;
    height: 500px;
}
#items-wrapper ul {
    padding: 0;
    margin: 0;
}
#items-wrapper li {
    height: 38px;
    font-size: 16px;
    list-style: none;
    cursor: pointer;
    text-align: center;
    padding-top: 10px;
}
#items-wrapper li span {
    margin-right: 15px;
    font-size: 14px;
}
#items-wrapper ul li:hover{
    background-color: #a5bff4;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
#content-wrapper .content-close{
    display: none;
}
#content-wrapper .content-open{
    display: block;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/backStageTemplate" %}

Setting width and height

You can customize the height and width of the backstage view using the height and width property. By default, dimensions are set based on the content added.

APP.VUE

```
<template>
  <ejs-ribbon ref="backstageRibbon"  :backStageMenu="backstageSettings">
    <e-ribbon-tabs>
      <e-ribbon-tab header="Home">
        <e-ribbon-groups>
          <e-ribbon-group header="Clipboard">
            <e-ribbon-collections>
              <e-ribbon-collection>
                <e-ribbon-items>
                  <e-ribbon-item type="Button" :buttonSettings="cutButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button" :buttonSettings="copyButton">
                  </e-ribbon-item>
                  <e-ribbon-item type="Button"
:buttonSettings="pasteButton">
                  </e-ribbon-item>
                </e-ribbon-items>
              </e-ribbon-collection>
            </e-ribbon-collections>
          </e-ribbon-group>
        </e-ribbon-groups>
      </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonBackstage } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
  provide: {
      ribbon: [RibbonBackstage]
  },
  data: function () {
      return {
        backstageSettings: {
          visible: true,
          items: [
```

```
            { id: 'home', text: 'Home', iconCss: 'e-icons e-home', content:
this.homeContentTemplate() },
            { id: 'new', text: 'New', iconCss: 'e-icons e-file-new',
content: this.newContentTemplate() },
            { id: 'open', text: 'Open', iconCss: 'e-icons e-folder-open',
content: this.openContentTemplate() },
          ],
          backButton: {
            text: 'Close',
          }
      },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton: { iconCss: "e-icons e-copy", content: "Copy"},
        pasteButton: { iconCss: "e-icons e-paste", content: "Paste"},
      };
    },
    methods: {
      homeContentTemplate() {
        return `
        <div id='home-wrapper' style="padding: 20px;">
          <div id='new-section' class='new-wrapper'>
              <div class='section-title'> New </div>
              <div class='category_container'>
                  <div class='doc_category_image'></div>
                  <span class='doc_category_text'> New document </span>
              </div>
          </div>
          <div id='block-wrapper'><div class='section-title'> Recent </div>
              <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                  <table>
                      <tbody>
                          <tr>
                              <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                              <td>
                                  <span style="display: block; font-size:
14px"> Ribbon.docx </span>
                                  <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
                              </td>
                          </tr>
                      </tbody>
                  </table>
              </div>
          </div>
      </div>
        `;
      },
      newContentTemplate() {
        return `
        <div id='new-content' style="padding: 20px;">
          <div id='new-section' class='new-wrapper'>
              <div class='section-title'> New </div>
              <div class='category_container'>
                  <div class='doc_category_image'></div>
                  <span class='doc_category_text'> New document </span>
```

```
                </div>
            </div>
        </div>
          `;
        },
        openContentTemplate() {
          return `
          <div style="padding: 20px;">
            <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                <table>
                    <tbody>
                        <tr>
                            <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                            <td>
                                <span style="display: block; font-size: 14px">
Open in Desktop App </span>
                                <span style="font-size: 12px"> Use the full
functionality of Ribbon </span>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
            <div class="section-content" style="padding: 12px 0px cursor:
pointer">
                <table>
                    <tbody>
                        <tr>
                            <td> <span class="doc_icon e-icons e-protect-
sheet"></span> </td>
                            <td>
                                <span style="display: block; font-size: 14px">
Protect Document </span>
                                <span style="font-size: 12px">To prevent
accidental changes,
                                    this document has been set to open as
view-only.</span>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
          `;
        },
    }
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
.section-title {
    font-size: 22px;
}
.new-docs {
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
}
.category_container {
    width: 150px;
    padding: 15px;
    text-align: center;
    cursor: pointer;
}
.doc_category_image {
    width: 80px;
    height: 100px;
    background-color: #fff;
    border: 1px solid rgb(125, 124, 124);
    text-align: center;
    overflow: hidden;
    margin: 0px auto 10px;
}
.doc_category_text {
    font-size: 16px;
}
.section-content {
    padding: 12px 0px;
    cursor: pointer;
}
.doc_icon {
    font-size: 16px;
    padding: 0px 10px;
}
.category_container:hover, .section-content:hover {
    background-color: #dfdfdf;
    border-radius: 5px;
    transition: all 0.3s;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/backstageCustomization" %}

Adding Backstage events

Help Pane

The help pane is dedicated area where the users can define help contents like controlling document permissions, sharing features, and more which appears on the right side of the Ribbon. You can use the helpPaneTemplate property to set the help pane contents.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <ejs-ribbon id="ribbon" helpPaneTemplate="undoredo">
    <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
            <e-ribbon-groups>
                <e-ribbon-group>
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="Button"
:buttonSettings="cutButton"></e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                    </e-ribbon-collections>
                </e-ribbon-group>
            </e-ribbon-groups>
        </e-ribbon-tab>
    </e-ribbon-tabs>
    <template v-slot:undoredo>
      <div class="helpPaneTemplate">
        <button class="action_btn"><label><span id="undo" class="e-icons e-
undo"></span> Undo </label></button>
        <button class="action_btn"><label><span id="redo" class="e-icons e-
redo"></span> Redo </label></button>
      </div>
    </template>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" }
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
  .action_btn {
    margin: 0px 3px;
    border: none;
    color: #ffffff;
    background-color: #0d6efd;
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  #undo, #redo{
    padding: 0px 3px ;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/helpPaneTemplate" %}

Tooltip

The Ribbon component supports tooltip to show additional information in the Ribbon items. The tooltip appears when the user hovers over a Ribbon item.

Adding title

You can use the title property to set the tooltip title for each Ribbon item.

`html

<template>

<ejs-ribbon id="ribbon">

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Clipboard">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="SplitButton" :splitButtonSettings="pasteButton"
:ribbonTooltipSettings="pasteOptions">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="Button" :buttonSettings="cutButton" :ribbonTooltipSettings="cutOptions">

</e-ribbon-item>

<e-ribbon-item type="Button" :buttonSettings="copyButton" :ribbonTooltipSettings="copyOptions">

</e-ribbon-item>

<e-ribbon-item type="Button" :buttonSettings="formatButton"
:ribbonTooltipSettings="formatOptions">

</e-ribbon-item>

</e-ribbon-items>

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
pasteButton:  {
iconCss: "e-icons e-paste",
content: "Paste",
items: [{ text: "Keep Source Format" }, { text: "Merge format" }, { text: "Keep text only" }]
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
formatButton:  { iconCss: "e-icons e-format-painter", content: "Format Painter" },
cutOptions: { title: "Cut" },
copyOptions: { title: "Copy" },
pasteOptions: { title: "Paste" },
formatOptions: { title: "Format Painter" }
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

Adding content

You can use the content property to set the tooltip content for each Ribbon item.

```
`html
<template>
<ejs-ribbon id="ribbon">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :splitButtonSettings="pasteButton"
:ribbonTooltipSettings="pasteOptions">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton" :ribbonTooltipSettings="cutOptions">
</e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="copyButton" :ribbonTooltipSettings="copyOptions">
</e-ribbon-item>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-item type="Button" :buttonSettings="formatButton"
:ribbonTooltipSettings="formatOptions">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
pasteButton:  {
iconCss: "e-icons e-paste",
content: "Paste",
items: [{ text: "Keep Source Format" }, { text: "Merge format" }, { text: "Keep text only" }]
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
formatButton:  { iconCss: "e-icons e-format-painter", content: "Format Painter" },
cutOptions: { title: "Cut", content: "Places the selected text or object on the clipboard so that you can
paste it somewhere else." },
copyOptions: { title: "Copy", content: "Copies the chosen text or object to the clipboard so that you can
reuse it elsewhere." },
pasteOptions: { title: "Paste", content: "Insert the clipboard content where the cursor is currently
placed." },
formatOptions: { title: "Format Painter", content: "Copies the formatting style of a selected text or
object and applies it to other content within the document." }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

Adding icon

You can use the iconCss property to specify the icons to be displayed in the tooltip.

```html
<template>
<ejs-ribbon id="ribbon">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :splitButtonSettings="pasteButton"
:ribbonTooltipSettings="pasteOptions">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
<e-ribbon-collection>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton" :ribbonTooltipSettings="cutOptions">
</e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="copyButton" :ribbonTooltipSettings="copyOptions">
</e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="formatButton"
:ribbonTooltipSettings="formatOptions">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
pasteButton:  {
iconCss: "e-icons e-paste",
content: "Paste",
items: [{ text: "Keep Source Format" }, { text: "Merge format" }, { text: "Keep text only" }]
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
formatButton:  { iconCss: "e-icons e-format-painter", content: "Format Painter" },
cutOptions: { title: "Cut", content: "Places the selected text or object on the clipboard so that you can
paste it somewhere else.", iconCss: "e-icons e-cut" },
```

copyOptions: { title: "Copy", content: "Copies the chosen text or object to the clipboard so that you can reuse it elsewhere.", iconCss: "e-icons e-copy" },

pasteOptions: { title: "Paste", content: "Insert the clipboard content where the cursor is currently placed.", iconCss: "e-icons e-paste" },

formatOptions: { title: "Format Painter", content: "Copies the formatting style of a selected text or object and applies it to other content within the document.", iconCss: "e-icons e-format-painter" }

};

}

};

</script>

<style>

@import "../node_modules/@syncfusion/ej2-base/styles/material.css";

@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";

@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

{% previewsample "page.domainurl/code-snippet/ribbon/Tooltip" %}

Customization

You can use the cssClass property to customize the appearance of the tooltip with your own custom styles.

APP.VUE

```
<template>
  <ejs-ribbon id="ribbon">
    <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
            <e-ribbon-groups>
                <e-ribbon-group header="Clipboard">
                    <e-ribbon-collections>
                      <e-ribbon-collection>
                        <e-ribbon-items>
                            <e-ribbon-item type="SplitButton"
:splitButtonSettings="pasteButton" :ribbonTooltipSettings="pasteOptions">
                            </e-ribbon-item>
                        </e-ribbon-items>
```

```
                        </e-ribbon-collection>
                        <e-ribbon-collection>
                          <e-ribbon-items>
                              <e-ribbon-item type="Button"
:buttonSettings="cutButton" :ribbonTooltipSettings="cutOptions">
                              </e-ribbon-item>
                              <e-ribbon-item type="Button"
:buttonSettings="copyButton" :ribbonTooltipSettings="copyOptions">
                              </e-ribbon-item>
                              <e-ribbon-item type="Button"
:buttonSettings="formatButton" :ribbonTooltipSettings="formatOptions">
                              </e-ribbon-item>
                          </e-ribbon-items>
                        </e-ribbon-collection>
                    </e-ribbon-collections>
                </e-ribbon-group>
            </e-ribbon-groups>
        </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        pasteButton:  {
        iconCss: "e-icons e-paste",
        content: "Paste",
        items: [{ text: "Keep Source Format" }, { text: "Merge format" }, {
text: "Keep text only" }]
        },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter" },

        cutOptions: { title: "Cut", content: "Places the selected text or
object on the clipboard so that you can paste it somewhere else.", cssClass:
"custom-tooltip" },
        copyOptions: { title: "Copy", content: "Copies the chosen text or
object to the clipboard so that you can reuse it elsewhere.", cssClass:
"custom-tooltip" },
        pasteOptions: { title: "Paste", content: "Insert the clipboard
content where the cursor is currently placed.", cssClass: "custom-tooltip"
},
        formatOptions: { title: "Format Painter", content: "Copies the
formatting style of a selected text or object and applies it to other
content within the document.", cssClass: "custom-tooltip" }
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
  :root {
    --borderColor: rgb(72, 72, 72);
    --black: #000000;
  }

  /* To customize the appearance of the tooltip */
  .custom-tooltip.e-ribbon-tooltip.e-popup {
    border: 2px solid var(--borderColor);
    border-radius: 5px;
    background: var(--black);
  }

  /* To customize the arrow of the tooltip */
  .custom-tooltip.e-ribbon-tooltip .e-arrow-tip .e-arrow-tip-inner.e-tip-
top,
  .custom-tooltip.e-ribbon-tooltip .e-arrow-tip .e-arrow-tip-inner.e-tip-
bottom {
    color: var(--black);;
  }
  .custom-tooltip.e-ribbon-tooltip .e-arrow-tip-outer.e-tip-top {
    border-bottom: 8px solid var(--borderColor);
  }
  .custom-tooltip.e-ribbon-tooltip .e-arrow-tip-outer.e-tip-bottom {
    border-top: 8px solid var(--borderColor);;
  }

  /* To change the size of the tooltip title */
  .custom-tooltip.e-ribbon-tooltip .e-tip-content .e-ribbon-tooltip-title {
    font-size: 14px;
  }

  /* To change the size of the tooltip content */
  .custom-tooltip.e-ribbon-tooltip .e-tip-content .e-ribbon-text-container
.e-ribbon-tooltip-content {
    font-size: 11px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/TooltipCustomization" %}

Ribbon Resizing

The Ribbon effectively resizes the ribbon elements while being resized. It extends when the ribbon size is increased and collapses when the ribbon size is decreased. The resizing can be performed in both the classic and simplified modes. Also, we have an option to resize the ribbon elements in the custom order.

Copyright © 2001 -2024 Syncfusion Inc.

In classic mode on resizing, the items size will be changed based on the available width of the tab content from the order of Large-> Medium-> Small and viceversa.

In simplified mode on resizing, the items size will be changed based on the available width of the tab content from the order of Medium-> Small and viceversa.

Defining items allowed size

You can use the allowedSizes property to maintain a constant size for an item. If allowedSizes is set, it keeps the size constant even when being resized.

APP.VUE

```
<template>
  <ejs-ribbon id="ribbon">
    <e-ribbon-tabs>
        <e-ribbon-tab header="Home">
            <e-ribbon-groups>
                <e-ribbon-group>
                    <e-ribbon-collections>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="SplitButton"
:allowedSizes="largeSize" :splitButtonSettings="pasteButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                        <e-ribbon-collection>
                            <e-ribbon-items>
                                <e-ribbon-item type="Button"
:allowedSizes="mediumSize" :buttonSettings="cutButton">
                                </e-ribbon-item>
                                <e-ribbon-item type="Button"
:allowedSizes="smallSize" :buttonSettings="copyButton">
                                </e-ribbon-item>
                            </e-ribbon-items>
                        </e-ribbon-collection>
                    </e-ribbon-collections>
                </e-ribbon-group>
            </e-ribbon-groups>
        </e-ribbon-tab>
    </e-ribbon-tabs>
  </ejs-ribbon>
</template>
<script>
  import Vue from "vue";
  import { RibbonItemSize, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
  Vue.use(RibbonPlugin);
  export default {
    data: function () {
      return {
        largeSize: RibbonItemSize.Large,
        mediumSize: RibbonItemSize.Medium,
        smallSize: RibbonItemSize.Small,
        pasteButton:{
          iconCss: "e-icons e-paste", content: "Paste",
          items: [{ text: "Keep Source Format" }, { text: "Merge format" },
{ text: "Keep text only" }]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        },
        cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
        copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
        formatButton:  { iconCss: "e-icons e-format-painter", content:
"Format Painter" }
      };
    }
  };
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/ribbon/itemsSize" %}

Defining items active size

You can use the activeSize property to define the item size intially, before it is being resized. When resized the activeSize property is updated based on the ribbon's overflow state, which is determined by the allowedSizes property being configured. By default, the value is Medium.

Events

This section describes the ribbon events that will be triggered when appropriate actions are performed. The following events are available in the ribbon component.

tabSelected

The tabSelected event is triggered after selecting the tab item.

`html

<template>

<ejs-ribbon :tabSelected="tabSelected">

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Clipboard">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="Button" :buttonSettings="cutButton">

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
};
},
methods:{
tabSelected: function(args){
// Your required action here
}
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

tabSelecting

The tabSelecting event is triggered before selecting the tab item.

`html

<template>

<ejs-ribbon :tabSelecting="tabSelecting">

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Clipboard">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="Button" :buttonSettings="cutButton">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

<script>

import Vue from "vue";

import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";

Vue.use(RibbonPlugin);

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
data: function () {
return {
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
};
},
methods:{
tabSelecting: function(args){
// Your required action here
}
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

ribbonCollapsing

The ribbonCollapsing event is triggered before collapsing the ribbon.

```
`html
<template>
<ejs-ribbon :ribbonCollapsing="ribbonCollapsing">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
};
},
methods:{
ribbonCollapsing: function(args){
// Your required action here
}
}
};
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

ribbonExpanding

The ribbonExpanding event is triggered before expanding the ribbon.

```
`html
<template>
<ejs-ribbon :ribbonExpanding="ribbonExpanding">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
};
},
methods:{
ribbonExpanding: function(args){
// Your required action here
}
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

launcherIconClick

The launcherIconClick event is triggered when the launcher icon of the group is clicked.

`html

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<ejs-ribbon :launcherIconClick="launchIconClick">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard" :showLauncherIcon="true">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
};
},
methods:{
launchIconClick: function(args){
// Your required action here
```

```
}
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

Button item events

clicked

The clicked event is triggered when the Button is clicked.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
cutButton: {
iconCss: "e-icons e-cut",
content: "Cut",
clicked: () => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

created

The created event is triggered when the Button is created.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Clipboard">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="Button" :buttonSettings="cutButton">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

<script>

import Vue from "vue";

import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";

Vue.use(RibbonPlugin);

export default {

data: function () {

return {

Copyright © 2001 -2024 Syncfusion Inc.

```
cutButton:  {
iconCss: "e-icons e-cut",
content: "Cut",
created: () => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

CheckBox item events

change

The change event is triggered when the Checkbox state is changed.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="CheckBox" :checkBoxSettings="rulerSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
rulerSettings: {
label: 'Ruler',
checked: false,
change: () => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

created

The created event is triggered once the Checkbox is created.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="CheckBox" :checkBoxSettings="rulerSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
rulerSettings: {
label: 'Ruler',
checked: false,
created: () => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

ColorPicker item events

change

The change event is triggered while changing the colors.

`html

```
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ColorPicker" :colorPickerSettings="colorPicker" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonColorPicker } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [ RibbonColorPicker ]
},
data: function () {
return {
colorPicker: {
value: "#123456",
change:(args: ChangeEventArgs) => {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
// Your required action here
}
},
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

created

The created event is triggered once the ColorPicker is created.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ColorPicker" :colorPickerSettings="colorPicker" >
</e-ribbon-item>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonColorPicker } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [ RibbonColorPicker ]
},
data: function () {
return {
colorPicker: {
value: "#123456",
created:(args: ChangeEventArgs) => {
// Your required action here
}
},
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

open

The open event is triggered while opening the ColorPicker popup.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ColorPicker" :colorPickerSettings="colorPicker" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { RibbonPlugin, RibbonColorPicker } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [ RibbonColorPicker ]
},
data: function () {
return {
colorPicker: {
value: "#123456",
open:(args: ChangeEventArgs) => {
// Your required action here
}
},
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

select

The select event is triggered while selecting the color in picker/palette, when showButtons property is enabled.

`html

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ColorPicker" :colorPickerSettings="colorPicker" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonColorPicker } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [ RibbonColorPicker ]
},
data: function () {
return {
colorPicker: {
value: "#123456",
select:(args: ChangeEventArgs) => {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
// Your required action here
}
},
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

beforeClose

The beforeClose event is triggered before closing the ColorPicker popup.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ColorPicker" :colorPickerSettings="colorPicker" >
</e-ribbon-item>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonColorPicker } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [ RibbonColorPicker ]
},
data: function () {
return {
colorPicker: {
value: "#123456",
beforeClose:(args: ChangeEventArgs) => {
// Your required action here
}
},
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

beforeOpen

The beforeOpen event is triggered before opening the ColorPicker popup.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Font">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="ColorPicker" :colorPickerSettings="colorPicker" >

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

<script>

import Vue from "vue";

Copyright © 2001 -2024 Syncfusion Inc.

```
import { RibbonPlugin, RibbonColorPicker } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [ RibbonColorPicker ]
},
data: function () {
return {
colorPicker: {
value: "#123456",
beforeOpen:(args: ChangeEventArgs) => {
// Your required action here
}
},
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

beforeTileRender

The beforeTileRender event is triggered while rendering each palette tile.

`html

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ColorPicker" :colorPickerSettings="colorPicker" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonColorPicker } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [ RibbonColorPicker ]
},
data: function () {
return {
colorPicker: {
value: "#123456",
beforeTileRender:(args: ChangeEventArgs) => {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
// Your required action here
}
},
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

ComboBox item events

change

The change event is triggered when an item in a popup is selected or when the model value is changed by the user.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ComboBox" :comboBoxSettings="styleOptions" >
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
styleOptions: {
dataSource: ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math', 'Courier New', 'Candara', 'Georgia'],
index: 3,
change: (args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

close

The close event is triggered when the popup is closed.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Font">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="ComboBox" :comboBoxSettings="styleOptions" >

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

<script>

import Vue from "vue";

import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
styleOptions: {
dataSource: ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math', 'Courier New', 'Candara', 'Georgia'],
index: 3,
close: (args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

open

The open event is triggered when the popup is opened.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ComboBox" :comboBoxSettings="styleOptions" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
styleOptions: {
dataSource: ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math', 'Courier New', 'Candara', 'Georgia'],
index: 3,
open: (args) => {
// Your required action here
}
}
};
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

created

The created event is triggered once the Combobox is created.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ComboBox" :comboBoxSettings="styleOptions" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
styleOptions: {
dataSource: ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math', 'Courier New', 'Candara', 'Georgia'],
index: 3,
created: (args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

`

filtering

The filtering event triggers on typing a character in the Combobox.

```html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="ComboBox" :comboBoxSettings="styleOptions" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
styleOptions: {
dataSource: ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math', 'Courier New', 'Candara', 'Georgia'],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
index: 3,
filtering: (args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

select

The select event is triggered when an item in the popup is selected.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Font">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-item type="ComboBox" :comboBoxSettings="styleOptions" >
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
styleOptions: {
dataSource: ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math', 'Courier New', 'Candara', 'Georgia'],
index: 3,
select: (args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

beforeOpen

The beforeOpen event triggers before opening the popup.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Font">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="ComboBox" :comboBoxSettings="styleOptions" >

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

<script>

import Vue from "vue";

Copyright © 2001 -2024 Syncfusion Inc.

```
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
styleOptions: {
dataSource: ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math', 'Courier New', 'Candara', 'Georgia'],
index: 3,
beforeOpen: (args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

DropDown item events

beforeClose

The beforeClose event is triggered before closing the DropdownButton popup.

```
`html
<template>
<ejs-ribbon>
```

```
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Header & Footer">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="DropDown" :dropDownSettings="tableSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
tableSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
beforeClose:(args) => {
// Your required action here
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

beforeOpen

The beforeOpen event is triggered before opening the Dropdown button popup.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Header & Footer">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="DropDown" :dropDownSettings="tableSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
tableSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
beforeOpen:(args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

beforeItemRender

The beforeItemRender event is triggered while rendering each popup item of the Dropdown button.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Header & Footer">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="DropDown" :dropDownSettings="tableSettings">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

<script>

import Vue from "vue";

import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";

Vue.use(RibbonPlugin);

export default {

data: function () {

Copyright © 2001 -2024 Syncfusion Inc.

```
return {
tableSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
beforeItemRender:(args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

open

The open event is triggered while opening the Dropdown button popup.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-group header="Header & Footer">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="DropDown" :dropDownSettings="tableSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
tableSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
open:(args) => {
// Your required action here
}
}
};
}
};
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

close

The close event is triggered while closing the Dropdown button popup.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Header & Footer">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="DropDown" :dropDownSettings="tableSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
tableSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
close:(args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

`

created

The created event is triggered when the DropDown is created.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Header & Footer">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="DropDown" :dropDownSettings="tableSettings">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

<script>

import Vue from "vue";

import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";

Vue.use(RibbonPlugin);

export default {

data: function () {

return {

tableSettings: {

content: 'Header',

Copyright © 2001 -2024 Syncfusion Inc.

```
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
created:(args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

select

The select event is triggered while selecting an action item in the Dropdown button popup.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Header & Footer">
<e-ribbon-collections>
<e-ribbon-collection>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-items>
<e-ribbon-item type="DropDown" :dropDownSettings="tableSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
tableSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
select:(args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";

@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

SplitButton item events

beforeClose

The beforeClose event is triggered before closing the SplitButton popup.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Header & Footer">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="SplitButton" :splitButtonSettings="dropDownSettings">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
dropDownSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
beforeClose:(args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

beforeOpen

The beforeOpen event is triggered before opening the SplitButton popup.

Copyright © 2001 -2024 Syncfusion Inc.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Header & Footer">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :splitButtonSettings="dropDownSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
dropDownSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
beforeOpen:(args) => {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

beforeItemRender

The beforeItemRender event is triggered while rendering each popup item of the SplitButton.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Header & Footer">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :splitButtonSettings="dropDownSettings">
</e-ribbon-item>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
dropDownSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
beforeItemRender:(args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

open

The open event is triggered while opening the SplitButton popup.

`html

<template>

<ejs-ribbon>

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Header & Footer">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="SplitButton" :splitButtonSettings="dropDownSettings">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

<script>

import Vue from "vue";

import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
dropDownSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
open:(args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

close

The close event is triggered while closing the SplitButton popup.

```
`html
<template>
<ejs-ribbon>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Header & Footer">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :splitButtonSettings="dropDownSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
dropDownSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
close:(args) => {
// Your required action here
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

created

The created event is triggered when the SplitButton is created.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Header & Footer">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :splitButtonSettings="dropDownSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
dropDownSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
created:(args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

select

The select event is triggered while selecting an action item in the SplitButton popup.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Header & Footer">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :splitButtonSettings="dropDownSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
return {
dropDownSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
select:(args) => {
// Your required action here
}
}
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

click

The click event is triggered while clicking the primary button in the SplitButton.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-ribbon-group header="Header & Footer">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="SplitButton" :splitButtonSettings="dropDownSettings">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
dropDownSettings: {
content: 'Header',
iconCss: 'e-icons e-header',
items: [{ text: 'Insert Header' }, { text: 'Edit Header' }, { text: 'Remove Header' }],
click:(args) => {
// Your required action here
}
}
};
}
};
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

GroupButton item events

beforeClick

The beforeClick event is triggered before selecting a button from the groupbutton items.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Paragraph">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="GroupButton" :allowedSizes="size"
:groupButtonSettings="groupButtonMultiple"></e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonItemSize, RibbonGroupButtonSelection, RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
size: RibbonItemSize.Medium,
groupButtonMultiple: {
selection: RibbonGroupButtonSelection.Multiple,
items: [{ iconCss: 'e-icons e-bold', content: 'Bold', selected: true, beforeClick: () => {
// Your required action here
} },
{iconCss: 'e-icons e-italic', content: 'Italic', beforeClick: () => {
// Your required action here
}},
{iconCss: 'e-icons e-underline', content: 'Underline', beforeClick: () => {
// Your required action here
}},
{iconCss: 'e-icons e-strikethrough', content: 'Strikethrough', beforeClick: () => {
// Your required action here
}},
{iconCss: 'e-icons e-change-case', content: 'Change Case', beforeClick: () => {
// Your required action here
}}
]
}
};
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

click

The click event is triggered when selecting a button from the groupbutton items.

```
`html
<template>
<ejs-ribbon>
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Paragraph">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="GroupButton" :allowedSizes="size"
:groupButtonSettings="groupButtonSingle"></e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonItemSize, RibbonGroupButtonSelection , RibbonPlugin } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
data: function () {
return {
size: RibbonItemSize.Medium,
groupButtonSingle: {
selection: RibbonGroupButtonSelection.Single,
items: [
{iconCss: 'e-icons e-align-left', content: 'Align Left', selected: true, click: () => {
// Your required action here
}},
{iconCss: 'e-icons e-align-center',content: 'Align Center', click: () => {
// Your required action here
}},
{iconCss: 'e-icons e-align-right',content: 'Align Right', click: () => {
// Your required action here
}},
{iconCss: 'e-icons e-justify',content: 'Justify', click: () => {
// Your required action here
}}
]
},
};
}
};
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

FileMenu events

beforeClose

The beforeClose event is triggered before closing the fileMenu popup.

```
`html
<template>
<ejs-ribbon :fileMenu="fileSettings">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton">
</e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="copyButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonFileMenu } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [RibbonFileMenu]
},
data: function () {
return {
fileSettings: {
visible: true,
menuItems: [
{ text: "New", iconCss: "e-icons e-file-new", id: "new" },
{ text: "Open", iconCss: "e-icons e-folder-open", id: "open" },
{ text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
{ text: "Save as", iconCss: "e-icons e-save", id: "save" }
],
beforeClose: (args) => {
// Your required action here
}
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
};
}
};
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

beforeOpen

The beforeOpen event is triggered before opening the fileMenu popup.

```
`html
<template>
<ejs-ribbon :fileMenu="fileSettings">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton">
</e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="copyButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
```

Copyright © 2001 -2024 Syncfusion Inc. 2018

```
</e-ribbon-tabs>
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonFileMenu } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [RibbonFileMenu]
},
data: function () {
return {
fileSettings: {
visible: true,
menuItems: [
{ text: "New", iconCss: "e-icons e-file-new", id: "new" },
{ text: "Open", iconCss: "e-icons e-folder-open", id: "open" },
{ text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
{ text: "Save as", iconCss: "e-icons e-save", id: "save" }
],
beforeOpen: (args) => {
// Your required action here
}
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";

@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

beforeItemRender

The beforeItemRender event is triggered while rendering each ribbon fileMenu item.

`html

<template>

<ejs-ribbon :fileMenu="fileSettings">

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Clipboard">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="Button" :buttonSettings="cutButton">

</e-ribbon-item>

<e-ribbon-item type="Button" :buttonSettings="copyButton">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

Copyright © 2001 -2024 Syncfusion Inc.

```
</ejs-ribbon>
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonFileMenu } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [RibbonFileMenu]
},
data: function () {
return {
fileSettings: {
visible: true,
menuItems: [
{ text: "New", iconCss: "e-icons e-file-new", id: "new" },
{ text: "Open", iconCss: "e-icons e-folder-open", id: "open" },
{ text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
{ text: "Save as", iconCss: "e-icons e-save", id: "save" }
],
beforeItemRender: (args) => {
// Your required action here
}
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc. 2021

```
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
</style>
`
```

open

The open event is triggered when the fileMenu popup is opened.

```
`html
<template>
<ejs-ribbon :fileMenu="fileSettings">
<e-ribbon-tabs>
<e-ribbon-tab header="Home">
<e-ribbon-groups>
<e-ribbon-group header="Clipboard">
<e-ribbon-collections>
<e-ribbon-collection>
<e-ribbon-items>
<e-ribbon-item type="Button" :buttonSettings="cutButton">
</e-ribbon-item>
<e-ribbon-item type="Button" :buttonSettings="copyButton">
</e-ribbon-item>
</e-ribbon-items>
</e-ribbon-collection>
</e-ribbon-collections>
</e-ribbon-group>
</e-ribbon-groups>
</e-ribbon-tab>
</e-ribbon-tabs>
</ejs-ribbon>
```

Copyright © 2001 -2024 Syncfusion Inc. 2022

```
</template>
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonFileMenu } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [RibbonFileMenu]
},
data: function () {
return {
fileSettings: {
visible: true,
menuItems: [
{ text: "New", iconCss: "e-icons e-file-new", id: "new" },
{ text: "Open", iconCss: "e-icons e-folder-open", id: "open" },
{ text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
{ text: "Save as", iconCss: "e-icons e-save", id: "save" }
],
open: (args) => {
// Your required action here
}
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc. 2023

@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

close

The close event is triggered when the fileMenu popup is closed.

`html

<template>

<ejs-ribbon :fileMenu="fileSettings">

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Clipboard">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="Button" :buttonSettings="cutButton">

</e-ribbon-item>

<e-ribbon-item type="Button" :buttonSettings="copyButton">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

Copyright © 2001 -2024 Syncfusion Inc. 2024

```
<script>
import Vue from "vue";
import { RibbonPlugin, RibbonFileMenu } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [RibbonFileMenu]
},
data: function () {
return {
fileSettings: {
visible: true,
menuItems: [
{ text: "New", iconCss: "e-icons e-file-new", id: "new" },
{ text: "Open", iconCss: "e-icons e-folder-open", id: "open" },
{ text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
{ text: "Save as", iconCss: "e-icons e-save", id: "save" }
],
close: (args) => {
// Your required action here
}
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc. 2025

@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

select

The select event is triggered while selecting an item in the ribbon fileMenu.

`html

<template>

<ejs-ribbon :fileMenu="fileSettings">

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Clipboard">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="Button" :buttonSettings="cutButton">

</e-ribbon-item>

<e-ribbon-item type="Button" :buttonSettings="copyButton">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

<script>

Copyright © 2001 -2024 Syncfusion Inc. 2026

```
import Vue from "vue";
import { RibbonPlugin, RibbonFileMenu } from "@syncfusion/ej2-vue-ribbon";
Vue.use(RibbonPlugin);
export default {
provide: {
ribbon: [RibbonFileMenu]
},
data: function () {
return {
fileSettings: {
visible: true,
menuItems: [
{ text: "New", iconCss: "e-icons e-file-new", id: "new" },
{ text: "Open", iconCss: "e-icons e-folder-open", id: "open" },
{ text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
{ text: "Save as", iconCss: "e-icons e-save", id: "save" }
],
select: (args) => {
// Your required action here
}
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
copyButton:  { iconCss: "e-icons e-copy", content: "Copy" },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";

@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";

</style>

`

Backstage view events

backStageItemClick

The backStageItemClick event is triggered when backstage item is selected.

`html

<template>

<ejs-ribbon :backStageMenu="backstageSettings">

<e-ribbon-tabs>

<e-ribbon-tab header="Home">

<e-ribbon-groups>

<e-ribbon-group header="Clipboard">

<e-ribbon-collections>

<e-ribbon-collection>

<e-ribbon-items>

<e-ribbon-item type="Button" :buttonSettings="cutButton">

</e-ribbon-item>

</e-ribbon-items>

</e-ribbon-collection>

</e-ribbon-collections>

</e-ribbon-group>

</e-ribbon-groups>

</e-ribbon-tab>

</e-ribbon-tabs>

</ejs-ribbon>

</template>

<script>

import Vue from "vue";

import { RibbonPlugin, RibbonBackstage } from "@syncfusion/ej2-vue-ribbon";

Vue.use(RibbonPlugin);

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
provide: {
ribbon: [RibbonBackstage]
},
data: function () {
return {
backstageSettings: {
visible: true,
items: [
{ id: 'home', text: 'Home', iconCss: 'e-icons e-home', content: '#homeContent', backStageItemClick: () =>
{
// Your required action here
}  },
],
backButton: {
text: 'Close',
}
},
cutButton:  { iconCss: "e-icons e-cut", content: "Cut" },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-ribbon/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

</style>
`

RichText Editor

Getting Started with the Vue Rich text editor Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Rich text editor component

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies are required to use the Rich Text Editor.

`js
|-- @syncfusion/ej2-vue-richtexteditor
|-- @syncfusion/ej2
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-buttons
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-splitbuttons
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-filemanager
|-- @syncfusion/ej2-vue-base
`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash
npm install -g @vue/cli
vue create quickstart
cd quickstart
npm run serve
`

or

Copyright © 2001 -2024 Syncfusion Inc.

`bash

yarn global add @vue/cli

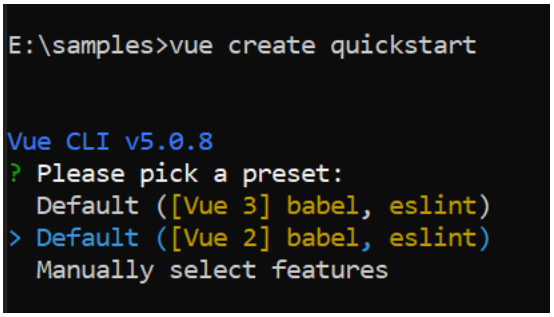
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Rich text editor component as an example. Install the @syncfusion/ej2-vue-richtexteditor package by running the following command:

`bash

npm install @syncfusion/ej2-vue-richtexteditor --save

`

or

`bash

yarn add @syncfusion/ej2-vue-richtexteditor

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Rich text editor component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Rich text editor component using:

1\. First, import and register the Rich text editor component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { RichTextEditorComponent, Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
export default {
components: {
'ejs-richtexteditor': RichTextEditorComponent
}
}
</script>
```

2\. In the template section, define the Rich text editor component with the height property.

~/SRC/APP.VUE

```
<template>
<ejs-richtexteditor ref="defaultRTE" :height="400">
<p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content.Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides IFRAME and DIV modes</p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for
developers.</p></li>
<li><p>Supports third-party library integration.</p></li>
<li><p>Allows preview of modified content before saving it.</p></li>
<li><p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p></li>
<li><p>Contains undo/redo manager.</p></li>
<li><p>Creates bulleted and numbered lists.</p></li>
</ul>
</ejs-richtexteditor>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

Initialize from `<div>` element

Rich Text Editor can be initialized on div element.

~/SRC/APP.VUE

```
<template>
  <div id="defaultRTE">
    <ejs-richtexteditor ref="defaultRTE" :height="340">
        <p>The Rich Text Editor component is WYSIWYG ("what you see is what
you get") editor that provides the best user experience to create and update
the content.Users can format their content using standard toolbar
commands.</p>
        <p><b>Key features:</b></p>
        <ul>
            <li><p>Provides IFRAME and DIV modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
        </ul>
    </ejs-richtexteditor>
  </div>
</template>
<script>
    import { RichTextEditorComponent, Toolbar, Link, Image, Count,
HtmlEditor, QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
    export default {
        components: {
            'ejs-richtexteditor': RichTextEditorComponent
        },
        provide: {
            richtexteditor:[Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar]
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs5" %}

Copyright © 2001 -2024 Syncfusion Inc.

Initialize from `<IFRAME>` element

The Rich Text Editor's content is placed in an iframe and isolated from the rest of the page.

Initialize the Rich Text Editor on div element and set the enable field of iframeSettings property as true.

~/SRC/APP.VUE

```
<template>
    <ejs-richtexteditor ref="defaultRTE" :height="350"
:iframeSettings="iframeConfig">
        <p>The Rich Text Editor component is WYSIWYG ("what you see is what
you get") editor that provides the best user experience to create and update
the content.Users can format their content using standard toolbar
commands.</p>
        <p><b>Key features:</b></p>
        <ul>
            <li><p>Provides IFRAME and DIV modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
        </ul>
    </ejs-richtexteditor>
</template>
<script>
    import { RichTextEditorComponent, Toolbar, Link, Image, Count,
HtmlEditor, QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
    export default {
        components: {
            'ejs-richtexteditor': RichTextEditorComponent
        },
        data() {
            return {
                iframeConfig: {
                    enable: true
                }
            }
        },
        provide: {
            richtexteditor:[Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar]
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs6" %}

Module injection

To create Rich Text Editor with additional features, inject the required modules. The following modules are used to extend Rich Text Editor's basic functionality.

- Toolbar - Inject this module to use Toolbar feature.
- Link - Inject this module to use link feature in Rich Text Editor.
- Image- Inject this module to use image feature in Rich Text Editor.
- Table- Inject this module to use table feature in Rich Text Editor.
- Count - Inject this module to use character count in Rich Text Editor.
- HtmlEditor - Inject this module to use Rich Text Editor as html editor.
- MarkdownEditor-Inject this module to use Rich Text Editor as markdown editor.
- QuickToolbar - Inject this module to use quick toolbar feature for the target element.
- Resize - Inject this module to use resize feature in Rich Text Editor.
- FileManager - Inject this module to use file browser feature in Rich Text Editor.
- PasteCleanup - Inject this module to use paste cleanup feature in Rich Text Editor.
- FormatPainter - Inject this module to use format painter feature in Rich Text Editor.
- EmojiPicker - Inject this module to use emoji picker feature in Rich Text Editor.

These modules should be injected into the Rich Text Editor through provide option.

Configure the toolbar

Configure the toolbar with the tools using items field of the toolbarSettings property as your application requires.

~/SRC/APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor :height="340"><p>The Rich Text Editor component
is WYSIWYG ("what you see is what you get") editor that provides the best
user experience to create and update the content. Users can format their
content using standard toolbar commands.</p>
        <p><b>Key features:</b></p>
          <ul>
            <li><p>Provides IFRAME and DIV modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
```

```
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
          </ul></ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
import { RichTextEditorComponent, Toolbar, HtmlEditor } from
"@syncfusion/ej2-vue-richtexteditor";
export default {
    components: {
      'ejs-richtexteditor': RichTextEditorComponent
    },
    provide:{
        richtexteditor:[Toolbar, HtmlEditor]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs7" %}

| and - can insert a vertical and horizontal separator lines in the toolbar.

Insert images and links

The image module inserts an image into Rich Text Editor's content area, and the link module links external resources such as website URLs, to selected text in the Rich Text Editor's content, respectively.

The link inject module adds a link icon to the toolbar and the image inject module adds an image icon to the toolbar.

Specifies the items to be rendered in the quick toolbar based on the target element such image, link and text element. The quick toolbar opens to customize the element by clicking the target element.

~/SRC/APP.VUE

```
<template>
<div>
<div class="control-section">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj"
:quickToolbarSettings="quickToolbarSettings" :height="340"
:toolbarSettings="toolbarSettings">
        <p>The Rich Text Editor is WYSIWYG ("what you see is what you get")
editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
        <a href="https://ej2.syncfusion.com/home/"
target="_blank">markdown</a>
        of the content</p>
        <p><b>Key features:</b></p>
          <ul>
          <li><p>Provides IFRAME and DIV modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
          </ul>
        </ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
import { RichTextEditorComponent, Toolbar, Image, Link, HtmlEditor,
QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
export default {
     components: {
        'ejs-richtexteditor': RichTextEditorComponent
     },
     data: function() {
        return {
        height: 400,
        quickToolbarSettings: {
            image: [
                'Replace', 'Align', 'Caption', 'Remove', 'InsertLink',
'OpenImageLink', '-',
                'EditImageLink', 'RemoveImageLink', 'Display', 'AltText',
'Dimension',
            ]
        },
        toolbarSettings: {
            items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
            'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
            'LowerCase', 'UpperCase', '|',
            'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
            'Outdent', 'Indent', '|',
            'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
```

```
            'SourceCode', 'FullScreen', '|', 'Undo', 'Redo'
          ]
        },
        };
    },
    provide:{
        richtexteditor:[Toolbar, Image, Link, HtmlEditor, QuickToolbar]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs8" %}

Retrieve the formatted content

To retrieve the editor contents, use value property of Rich Text Editor.

`ts

var rteValue = this.$refs.rteObj.ej2Instances.value;

`

Or, you can use the public method, getHtml to retrieve the Rich Text Editor content.

`ts

var rteValue = this.$refs.rteObj.ej2Instances.getHtml();

`

To fetch the Rich Text Editor's text content, use getText method of Rich Text Editor.

`ts

var rteValue = this.$refs.rteObj.ej2Instances.getText();

`

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

Copyright © 2001 -2024 Syncfusion Inc.

`bash

yarn run serve

`

~/SRC/APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj"
:quickToolbarSettings="quickToolbarSettings" :height="340"
:toolbarSettings="toolbarSettings">
        <p>The Rich Text Editor is WYSIWYG ("what you see is what you get")
editor useful to create and edit content,and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
        <a href="https://ej2.syncfusion.com/home/"
target="_blank">markdown</a>
        of the content</p>
        <p><b>Key features:</b></p>
          <ul><li><p>Provides IFRAME and DIV modes</p></li>
          <li><p>Capable of handling markdown editing.</p></li>
          <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
          <li><p>Provides a fully customizable toolbar.</p></li>
          <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
          <li><p>Supports third-party library integration.</p></li>
          <li><p>Allows preview of modified content before saving
it.</p></li>
          <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
          </ul></ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
import { RichTextEditorComponent, Toolbar, Image, Link, HtmlEditor,
QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
export default {
     components: {
        'ejs-richtexteditor': RichTextEditorComponent
     },
     data: function() {
        return {
        height: 400,
        quickToolbarSettings: {
            image: [
                'Replace', 'Align', 'Caption', 'Remove', 'InsertLink',
'OpenImageLink', '-',
                'EditImageLink', 'RemoveImageLink', 'Display', 'AltText',
'Dimension',
            ],
```

```
            link: ['Open', 'Edit', 'UnLink']
        },
        toolbarSettings: {
            items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
            'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
            'LowerCase', 'UpperCase', '|',
            'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
            'Outdent', 'Indent', '|',
            'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
            'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
        },
        };
    },
    provide:{
        richtexteditor:[Toolbar, Image, Link, HtmlEditor, QuickToolbar]
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs9" %}

Add content using valueTemplate property

Rich Text Editor content can be added using valueTemplate property as given below

~/SRC/APP.VUE

```
<template>
  <div id="defaultRTE">
    <ejs-richtexteditor :valueTemplate="'vueTemplate'"  ref="defaultRTE"
:height="340">
    <template v-slot:vueTemplate>
        <p>The Rich Text Editor component is WYSIWYG ("what you see is what
you get") editor that provides the best user experience to create and update
the content.Users can format their content using standard toolbar
commands.</p>
    </template>
    </ejs-richtexteditor>
  </div>
</template>
<script>
    import { RichTextEditorComponent, Toolbar, Link, Image, Count,
HtmlEditor, QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';

    export default {
         components: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            'ejs-richtexteditor': RichTextEditorComponent
          },
          data: function() {
    return { };
          },
          provide: {
              richtexteditor:[Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar]
          }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs10" %}

See Also

- How to change the editor type
- How to render the iframe
- How to render the toolbar in inline mode

You can refer to our Vue Rich Text Editor feature tour page for its groundbreaking feature representations. You can also explore our Vue Rich Text Editor example that shows how to render and configure the rich text editor tools.

Getting Started with the Vue richtexteditor Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Rich Text Editor component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

Copyright © 2001 -2024 Syncfusion Inc.

`bash
npm create vite@latest
`

or

`bash
yarn create vite
`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1. Define the project name: We can specify the name of the project directly. Let's specify the name of the project as `my-project` for this article.

`bash
? Project name: » my-project
`

2. Select `Vue` as the framework. It will create a Vue 3 project.

`bash
? Select a framework: » - Use arrow-keys. Return to submit.
Vanilla
Vue
React
Preact
Lit
Svelte
Others
`

3. Choose `JavaScript` as the framework variant to build this Vite project using JavaScript and Vue.

`bash
? Select a variant: » - Use arrow-keys. Return to submit.
JavaScript
TypeScript
Customize with create-vue ↗

Nuxt ↗
`

4. Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash
cd my-project
npm install
`

or

`bash
cd my-project
yarn install
`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Rich Text Editor component as an example. To use the Vue Rich Text Editor component in the project, the @syncfusion/ej2-vue-richtexteditor package needs to be installed using the following command:

`bash
npm install @syncfusion/ej2-vue-richtexteditor --save
`

or

`bash
yarn add @syncfusion/ej2-vue-richtexteditor
`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Rich Text Editor component and its dependents were imported into the <style> section of **src/App.vue** file.

Copyright © 2001 -2024 Syncfusion Inc.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue Rich Text Editor component using Composition API or Options API:

1. First, import and register the Rich Text Editor component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { RichTextEditorComponent as
EjsRichtexteditor,Toolbar,Link,Image,HtmlEditor } from "@syncfusion/ej2-vue-
richtexteditor";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { RichTextEditorComponent,Toolbar,Link,Image,HtmlEditor } from
"@syncfusion/ej2-vue-richtexteditor";
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div>
<div class="control-section">
<div class="sample-container">
<div class="default-section">
<ejs-richtexteditor id="default" ref="rteInstance" v-bind:value="value"
></ejs-richtexteditor>
</div>
</div>
</div>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script setup>
import { RichTextEditorComponent as
EjsRichtexteditor,Toolbar,Link,Image,HtmlEditor } from "@syncfusion/ej2-vue-
richtexteditor";
import { provide,ref } from "vue";
const rteInstance = ref(null);
const value = ref("<p>The Rich Text Editor component is a WYSIWYG editor
that provides the best user experience to create and update the content.
Users can format their content using standard toolbar commands.</p>");
const richtexteditor =  [Toolbar, Link, Image, HtmlEditor];
provide('richtexteditor', richtexteditor);
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div>
<div class="control-section">
<div class="sample-container">
<div class="default-section">
<ejs-richtexteditor ref="rte_instance" id="rte1" v-bind:value="value"></ejs-
richtexteditor>
</div>
</div>
</div>
</div>
</template>
<script>
import { RichTextEditorComponent,Toolbar,Link,Image,HtmlEditor } from
"@syncfusion/ej2-vue-richtexteditor";
export default {
name: "NormalView",
components: {
"ejs-richtexteditor": RichTextEditorComponent
},
data: function () {
return {
value: "<p>The Rich Text Editor component is a WYSIWYG editor that provides
the best user experience to create and update the content. Users can format
their content using standard toolbar commands.</p>"
}
},
provide: {
richtexteditor: [Toolbar, Link, Image, HtmlEditor]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
};
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

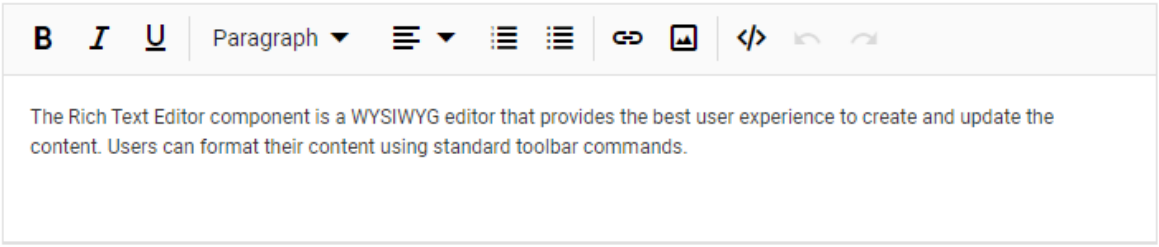
or

`bash

yarn run dev

`

The output will appear as follows:



Sample: vue-3-richrexteditor-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

Editor modes in Vue Rich text editor component

The Rich Text Editor component used to create and edit the content and return valid HTML markup or markdown (MD) of the content. It supports the following two editing formation.

- HTML editor
- Markdown editor

Copyright © 2001 -2024 Syncfusion Inc.

HTML editor

Rich Text Editor is a WYSIWYG editing control for formatting the word content as HTML.

The HTML editing mode is the default mode in Rich Text Editor to format the content through the available toolbar items to return the valid HTML markup. Set the editorMode property as **HTML**.

To create Rich Text Editor with HTML editing feature, inject the HtmlEditor module to the Rich Text Editor using the RichTextEditor.Inject(HtmlEditor) method.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs2" %}

Markdown editor

Set the editorMode property as **Markdown**, to create or edit the content and apply formatting to view markdown formatted content.

The third-party library such as Marked or any other library is used to convert markdown into HTML content.

- Supported tags are h6, h5, h4, h3, h2, h1, blockquote, pre, p, OL, and UL.
- Supported selection tags are Bold, Italic, StrikeThrough, InlineCode, SubScript, SuperScript, UpperCase, and LowerCase.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="rteInstance" :height="350" editorMode="Markdown"
:value="value" :toolbarSettings="toolbarConfig" :created="created">
  </ejs-richtexteditor>
</template>
<style>
  .e-richtexteditor .e-rte-content .e-content{
    min-height: 150px;
  }
  .e-richtexteditor .e-rte-content textarea.e-content {
    float: left;
    border-right: 1px solid rgba(0, 0, 0, 0.12);
  }
  .e-richtexteditor .e-rte-content {
    overflow: hidden;
  }
  .e-md-preview::before {
    content: '\e345';
  }
  .e-rte-content .e-content.e-pre-source {
    width: 100%;
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  .e-icon-btn.e-active .e-md-preview.e-icons::before {
    content: '\e350';
  }
</style>
<script>
  import Vue from 'vue';
  import { createElement, KeyboardEventArgs } from "@syncfusion/ej2-base";
  import { RichTextEditorPlugin, Toolbar, Link, Image, MarkdownEditor } from
'@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        value: `***Overview***
  The Rich Text Editor component is WYSIWYG ("what you see is what you get")
editor used to create and edit the content and return valid HTML markup or
markdown (MD) of the content. The editor provides a standard toolbar to
format content using its commands. Modular library features to load the
necessary functionality on demand. The toolbar contains commands to align
the text, insert link, insert image, insert list, undo/redo operation, HTML
view, and more.
  ***Key features***
  - *Mode*: Provides IFRAME and DIV mode.
  - *Module*: Modular library to load the necessary functionality on demand.
  - *Toolbar*: Provide a fully customizable toolbar.
  - *Editing*: HTML view to edit the source directly for developers.
  - *Third-party Integration*: Supports to integrate third-party library.
  - *Preview*: Preview the modified content before saving it.
  - *Tools*: Handling images, hyperlinks, video, uploads and more.
  - *Undo and Redo*: Undo/redo manager.
  - *Lists*:Creates bulleted and numbered list.`,
        toolbarConfig: {
          items: [
            'Bold', 'Italic', 'StrikeThrough', '|',
            'Formats', 'OrderedList', 'UnorderedList', '|',
            'CreateLink', 'Image', '|',
            {
              tooltipText: 'Preview',
              template: '<button id="preview-code" class="e-tbar-btn e-
control e-btn e-icon-btn">' +
                '<span class="e-btn-icon e-md-preview e-
icons"></span></button>'
            },
            '|', 'Undo', 'Redo'
          ]
        }
      }
    },
    methods: {
      created: function() {
        this.textArea =
this.$refs.rteInstance.$el.ej2_instances[0].element.querySelector('.e-
content');
        this.textArea.onkeyup = (Event) => {
          this.markDownConversion();
        };
        document.getElementById('preview-code').onclick = () => {
```

```
            this.fullPreview({ mode: true, type: 'preview'});
            if (event.target.parentElement.classList.contains('e-active')) {
              this.$refs.rteInstance.ej2Instances.disableToolbarItem([
                'Bold', 'Italic', 'StrikeThrough', 'Formats', 'OrderedList',
                'UnorderedList', 'CreateLink', 'Image'
              ]);

event.target.parentElement.parentElement.nextElementSibling.classList.add('e
-overlay');
            } else {
              this.$refs.rteInstance.ej2Instances.enableToolbarItem([
                'Bold', 'Italic', 'StrikeThrough', 'Formats', 'OrderedList',
                'UnorderedList', 'CreateLink', 'Image'
              ]);

event.target.parentElement.parentElement.nextElementSibling.classList.remove
('e-overlay');
            }
          };
        },
        markDownConversion: function() {
          if (document.getElementById('preview-code').classList.contains('e-
active')) {
            var id =  this.$refs.rteInstance.ej2Instance.getID() + 'html-
view';
            var htmlPreview =
this.$refs.rteInstance.$el.ej2_instances[0].element.querySelector('#' + id);
            htmlPreview.innerHTML = marked.parse(this.textArea.value);
          }
        },
        fullPreview: function(event){
          var mdSource = document.getElementById('preview-code');
          var id = this.$refs.rteInstance.ej2Instance.getID() + 'html-view';
          var htmlPreview =
this.$refs.rteInstance.$el.ej2_instances[0].element.querySelector('#' + id);
          if ((mdSource.classList.contains('e-active')) && event.mode) {
            mdSource.classList.remove('e-active');
            this.textArea.style.display = 'block';
            this.textArea.style.width = '100%';
            htmlPreview.style.display = 'none';
          } else {
            mdSource.classList.add('e-active');
            if (!htmlPreview) {
              htmlPreview = document.createElement('div');
              htmlPreview.id = id;
              htmlPreview.setAttribute('class', 'e-content');
              this.textArea.parentNode.appendChild(htmlPreview);
            }
            if (event.type === 'preview') {
              this.textArea.style.display = 'none';
              htmlPreview.classList.add('e-pre-source');
            } else {
              htmlPreview.classList.remove('e-pre-source');
              this.textArea.style.width = '50%';
            }
            htmlPreview.style.display = 'block';
```

```
            htmlPreview.innerHTML =
marked.parse(this.$refs.rteInstance.ej2Instances.contentModule.getEditPanel(
).value);
            mdSource.parentElement.title = 'Code View';
          }
        }
      },
      provide: {
        richtexteditor: [Toolbar, Link, Image, MarkdownEditor]
      }
    }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs1" %}

To create Rich Text Editor with Markdown editing feature, inject the MarkdownEditor module to the Rich Text Editor using the RichTextEditor.Inject(MarkdownEditor) method.

For further details on Markdown editing, refer to the Markdown section.

See Also

- How to integrate the third party library
- How to render the iframe

Toolbar in Vue Rich text editor component

The Rich Text Editor toolbar contains a collection of tools such as bold, italic, and text alignment buttons that are used to format the content. However, in most integrations, you can customize the toolbar configurations easily to suit your needs.

To create Rich Text Editor with Markdown editing feature, inject the Toolbar module to the Rich Text Editor using the RichTextEditor.Inject(Toolbar) method.

The Rich Text Editor allows you to configure different types of toolbar using toolbarSettings.type property. The types of toolbar are:

1. Expand
2. MultiRow

Copyright © 2001 -2024 Syncfusion Inc.

Expand Toolbar

The default mode of toolbarSettings.type is Expand, it will hide the overflowing items in the next row. By clicking the expand arrow, view the overflowing toolbar items.

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj" :height="340"
:toolbarSettings="toolbarSettings"><p>The Rich Text Editor component is
WYSIWYG ("what you see is what you get") editor that provides the best user
experience to create and update the content. Users can format their content
using standard toolbar commands.</p>
        <p><b>Key features:</b></p>
          <ul>
            <li><p>Provides IFRAME and DIV modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
          </ul></ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, HtmlEditor } from "@syncfusion/ej2-
vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
        toolbarSettings: {
            type: 'Expand',
            items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
            'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
            'LowerCase', 'UpperCase', '|',
            'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
            'Outdent', 'Indent', '|',
            'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
            'SourceCode', 'FullScreen', '|', 'Undo', 'Redo'
          ]
        },
        };
    },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    provide:{
        richtexteditor:[Toolbar, HtmlEditor]
    }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs16" %}

Multi-row Toolbar

Set the type as MultiRow in toolbarSettings to hide the overflowing items in the next row. All toolbar items are visible.

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj" :height="340"
:toolbarSettings="toolbarSettings"><p>The Rich Text Editor component is
WYSIWYG ("what you see is what you get") editor that provides the best user
experience to create and update the content. Users can format their content
using standard toolbar commands.</p>
        <p><b>Key features:</b></p>
          <ul>
            <li><p>Provides IFRAME and DIV modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
          </ul></ejs-richtexteditor>
        </div>
    </div>
```

```
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, HtmlEditor } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
        toolbarSettings: {
            type: 'MultiRow',
            items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
            'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
            'LowerCase', 'UpperCase', '|',
            'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
            'Outdent', 'Indent', '|',
            'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
            'SourceCode', 'FullScreen', '|', 'Undo', 'Redo'
          ]
        },
        };
    },
    provide:{
        richtexteditor:[Toolbar, HtmlEditor]
    }
}
</script>
<style>
@import "https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs17" %}

Floating Toolbar

By default, toolbar is float at the top of the Rich Text Editor on scrolling. It can be customized by specifying the offset of the floating toolbar from documents top position using floatingToolbarOffset.

Enable or disable the floating toolbar using enableFloating of thetoolbarSettings property.

APP.VUE

```
<template>
```

```
    <div>
        <div class="col-lg-4 property-section">
            <div title="Properties" id="property">
                <table title="Properties" id="property">
                    <tbody>
                        <tr>
                            <td>
                                <div>
                                    <ejs-checkbox label='Enable Floating'
ref="checkInstance" :change="onFloatChange" id="float"
:checked="false"></ejs-checkbox>
                                </div>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
        <div class="control-section">
            <div class="sample-container">
                <div class="default-section">
                    <ejs-richtexteditor ref="rteInstance" :height="340"
:toolbarSettings="toolbarSettings"><p>The Rich Text Editor component is
WYSIWYG ("what you see is what you get") editor that provides the best user
experience to create and update the content. Users can format their content
using standard toolbar commands.</p>
                        <p><b>Key features:</b></p>
                        <ul>
                            <li><p>Provides IFRAME and DIV modes</p></li>
                            <li><p>Capable of handling markdown
editing.</p></li>
                            <li><p>Contains a modular library to load the
necessary functionality on demand.</p></li>
                            <li><p>Provides a fully customizable
toolbar.</p></li>
                            <li><p>Provides HTML view to edit the source
directly for developers.</p></li>
                            <li><p>Supports third-party library
integration.</p></li>
                            <li><p>Allows preview of modified content before
saving it.</p></li>
                            <li><p>Handles images, hyperlinks, video,
hyperlinks, uploads, etc.</p></li>
                        </ul>
                    </ejs-richtexteditor>
                </div>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, HtmlEditor } from "@syncfusion/ej2-
vue-richtexteditor";
import { CheckBoxPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(RichTextEditorPlugin);
Vue.use(CheckBoxPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
    data: function() {
        return {
        toolbarSettings: {
            enableFloating: false
        },
        };
    },
        methods: {
        onFloatChange: function(args) {

this.$refs.rteInstance.ej2Instances.toolbarSettings.enableFloating =
args.checked;
            this.$refs.rteInstance.dataBind();
        }
    },
    provide:{
        richtexteditor:[Toolbar, HtmlEditor]
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs18" %}

Toolbar items

The following table lists the tools available in the toolbar.

| **Name** | **Summary** | **Initialization** |

| --- | --- | --- |

| Undo | Allows to undo the actions. | toolbarSettings: {
 items: ['Undo']
 } |

| Redo | Allows to redo the actions. | toolbarSettings: {
 items: ['Redo']
 } |

| Alignment | Align the content with left, center, and right margin. | toolbarSettings: {
 items: ['Alignments']
 } |

| OrderedList | Create a new list item (numbered). | toolbarSettings: {
 items: ['OrderedList']
 } |

| UnorderedList | Create a new list item (bulleted). | toolbarSettings: {
 items: ['UnorderedList']
 } |

Copyright © 2001 -2024 Syncfusion Inc.

| Indent |Allows to increase the indent level of the content. | toolbarSettings: {
 items: ['Indent']
 } |

| Outdent | Allows to decrease the indent level of the content. | toolbarSettings: {
 items: ['Outdent']
 } |

| Hyperlink | Creates a hyperlink to a text or image to a specific location in the content. | toolbarSettings: {
 items: ['CreateLink']
 } |

| Images | Inserts an image from an online source or local computer. | toolbarSettings: {
 items: ['Image']
 } |

| LowerCase | Change the case of selected text to lower in the content. | toolbarSettings: {
 items: ['LowerCase']
 } |

| UpperCase | Change the case of selected text to upper in the content. | toolbarSettings: {
 items: ['UpperCase']
 } |

| SubScript | Makes the selected text as subscript (lower). | toolbarSettings: {
 items: ['SubScript']
 } |

| SuperScript | Makes the selected text as superscript (higher). | toolbarSettings: {
 items: ['SuperScript']
 } |

| Print | Allows to print the editor content. | toolbarSettings: {
 items: ['Print']
 } |

| FontName | Defines the fonts that appear under the Font Family DropDownList from the Rich Text Editor's toolbar. | toolbarSettings: {
 items: ['FontName']
 } |

| FontSize | Defines the font sizes that appear under the Font Size DropDownList from the Rich Text Editor's toolbar. | toolbarSettings: {
 items: ['FontSize']
 } |

| FontColor | Specifies an array of colors can be used in the colors pop-up for font color. | toolbarSettings: {
 items: ['FontColor']
 } |

| BackgroundColor | Specifies an array of colors can be used in the colors pop-up for background color. | toolbarSettings: {
 items: ['BackgroundColor']
 } |

| Format | An object with the options that will appear in the paragraph format drop-down from the toolbar. | toolbarSettings: {
 items: ['Formats']
 } |

| StrikeThrough | Apply double line strike through formatting for the selected text. | toolbarSettings: {
 items: ['StrikeThrough']
 } |

| ClearFormat | The clear format tool is used to remove all formatting styles (such as bold, italic, underline, color, superscript, subscript, and more) from currently selected text. As a result, all the formatting text will be cleared and return to its default formatting styles. | toolbarSettings: {
 items: ['ClearFormat']
 } |

| FullScreen | Stretches the editor to the maximum width and height of the browser window. | toolbarSettings: {
 items: ['FullScreen']
 } |

| SourceCode | The RichTextBox includes the ability for users to directly edit the HTML code via Source View. If you made any modification in source view directly, synchronize with design view. | toolbarSettings: {
 items: ['SourceCode']
 } |

| NumberFormatList | Allows to create list items with various list style types(numbered)|toolbarSettings: {
 items: ['NumberFormatList']
 } |

Copyright © 2001 -2024 Syncfusion Inc.

| BulletFormatList | Allows to create list items with various list style types(bulleted)|toolbarSettings: {
 items: ['BulletFormatList']
 } |

By default, tools will be arranged in following order.

items: ['Bold', 'Italic', 'Underline', '|', 'Formats', 'Alignments', 'OrderedList', 'UnorderedList', '|', 'CreateLink', 'Image', '|', 'SourceCode', 'Undo', 'Redo']

The tools order can be customized as our application requirement. If you are not specifying any tools order, the editor will create the toolbar with default items.

Custom tool

The Rich Text Editor allows you to configure your own commands to its toolbar using the toolbarSettings property. The command can be plain text, icon, or HTML template. The order and the group can also be defined where the command should be included. Bind the action to the command by getting its instance.

This sample shows how to add your own commands to the toolbar of the Rich Text Editor. The **Ω** command is added to insert special characters in the editor. By clicking the **Ω** command, it will show the special characters list, and then choose the character to be inserted in the editor.

The following code snippet illustrates custom tool with tooltip text which will be included in items field of the toolbarSettings property.

In the following sample, once Rich Text Editor control is created, the concern event will be created the Dialog component can be rendered and target as Rich Text Editor content.

`javascript

{

tooltipText: 'Insert Symbol',

undo: true,

click: this.onClick.bind(this),

template: '<button class="e-tbar-btn e-btn" tabindex="-1" id="custom_tbar" style="width:100%"> Ω </button>'

}

`

Click the **Ω** command to show the special characters list, and then choose the character to be inserted in the editor.

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section" id="rteSection" style="min-height:
360px;">
        <ejs-richtexteditor ref="customObj"
:toolbarSettings="toolbarSettings" :created="onCreate"><p style="margin-
right:10px">The custom command "insert special character" is configured as
```

Copyright © 2001 -2024 Syncfusion Inc.

```
the last item of the toolbar. Click on the command and choose the special
character you want to include from the popup.</p></ejs-richtexteditor>
        <ejs-dialog id='rteDialog' :buttons='dlgButtons' :width='width'
:height="height" :header='header' ref="dialogObj"
:overlayClick='dialogOverlay' :visible='visible'
:showCloseIcon='showCloseIcon' :isModal='modal' :cssClass="cssClass"
target='#rteSection' :created="dialogCreate">
        </ejs-dialog>
        <div id="customTbarDialog" style="display: none">
                <div id="rteSpecial_char">
                   <div class="char_block" title="&#94;">&#94;</div>
                    <div class="char_block" title="&#95;">&#95;</div>
                    <div class="char_block" title="&#96;">&#96;</div>
                    <div class="char_block" title="&#123;">&#123;</div>
                    <div class="char_block" title="&#124;">&#124;</div>
                    <div class="char_block" title="&#125;">&#125;</div>
                    <div class="char_block" title="&#126;">&#126;</div>
                    <div class="char_block" title=" "> </div>
                    <div class="char_block" title="&#161;">&#161;</div>
                    <div class="char_block" title="&#162;">&#162;</div>
                    <div class="char_block" title="&#163;">&#163;</div>
                    <div class="char_block" title="&#164;">&#164;</div>
                    <div class="char_block" title="&#165;">&#165;</div>
                    <div class="char_block" title="&#x20B9;">&#x20B9;</div>
                    <div class="char_block" title="&#166;">&#166;</div>
                    <div class="char_block" title="&#167;">&#167;</div>
                    <div class="char_block" title="&#168;">&#168;</div>
                    <div class="char_block" title="&#169;">&#169;</div>
                    <div class="char_block" title="&#170;">&#170;</div>
                    <div class="char_block" title="&#171;">&#171;</div>
                    <div class="char_block" title="&#172;">&#172;</div>
                    <div class="char_block" title="&#173;">&#173;</div>
                    <div class="char_block" title="&#174;">&#174;</div>
                    <div class="char_block" title="&#175;">&#175;</div>
                    <div class="char_block" title="&#176;">&#176;</div>
                    <div class="char_block" title="&#177;">&#177;</div>
                    <div class="char_block" title="&#178;">&#178;</div>
                    <div class="char_block" title="&#179;">&#179;</div>
                    <div class="char_block" title="&#180;">&#180;</div>
                    <div class="char_block" title="&#181;">&#181;</div>
                    <div class="char_block" title="&#182;">&#182;</div>
                    <div class="char_block" title="&#183;">&#183;</div>
                    <div class="char_block" title="&#184;">&#184;</div>
                    <div class="char_block" title="&#185;">&#185;</div>
                    <div class="char_block" title="&#186;">&#186;</div>
                    <div class="char_block" title="&#187;">&#187;</div>
                    <div class="char_block" title="&#188;">&#188;</div>
                    <div class="char_block" title="&#189;">&#189;</div>
                    <div class="char_block" title="&#190;">&#190;</div>
                    <div class="char_block" title="&#191;">&#191;</div>
                    <div class="char_block" title="&#192;">&#192;</div>
                    <div class="char_block" title="&#193;">&#193;</div>
                    <div class="char_block" title="&#194;">&#194;</div>
                    <div class="char_block" title="&#195;">&#195;</div>
                </div>
            </div>
        </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
</div>
</div>
</template>
<style>
    #rteSpecial_char .char_block {
        display: inline-block;
    }
    #custom_tbar,
    #custom_tbar div{
        cursor: pointer;
        font-size: 16px;
    }
    #rteSpecial_char {
        padding: 15px 0 15px 0;
    }
    #rteSpecial_char .char_block.e-active {
        outline: 1px solid #e3165b;
        border-color: #e3165b;
    }
    #rteSpecial_char .char_block {
        width: 30px;
        height: 30px;
        line-height: 30px;
        margin: 0 5px 5px 0;
        text-align: center;
        vertical-align: middle;
        border: 1px solid #DDDDDD;
        font-size: 20px;
        cursor: pointer;
        user-select: none;
    }
</style>
<script>
import Vue from "vue";
import { Browser, addClass, removeClass } from "@syncfusion/ej2-base";
import { RichTextEditorPlugin, Toolbar, Link, NodeSelection, Image,
QuickToolbar, HtmlEditor } from "@syncfusion/ej2-vue-richtexteditor";
import { DialogPlugin } from '@syncfusion/ej2-vue-popups';
let proxy = undefined;
Vue.use(RichTextEditorPlugin);
Vue.use(DialogPlugin);
export default {
    data: function() {
        return {
            selection: new NodeSelection(),
            ranges: null,
            header: 'Special Characters',
            visible: false,
            modal: true,
            showCloseIcon: false,
            width: '520px',
            height: '310px',
            dlgButtons: [{ click: this.onInsert.bind(this), buttonModel: {
isPrimary:'true', content: 'Insert' } }, { buttonModel: { content: 'Cancel'
}, click: this.dialogOverlay.bind(this) }],
            toolbarSettings: {
```

```
                items: ['Bold', 'Italic', 'Underline', '|', 'Formats',
'Alignments', 'OrderedList',
                'UnorderedList', '|', 'CreateLink', 'Image', '|',
'SourceCode',
                {
                    tooltipText: 'Insert Symbol',
                    undo: true,
                    click: this.onClick.bind(this),
                    template: '<button class="e-tbar-btn e-btn" tabindex="-
1" id="custom_tbar"  style="width:100%"><div class="e-tbar-btn-text"
style="font-weight: 500;"> &#937;</div></button>'
                }, '|', 'Undo', 'Redo']
            },
            cssClass: "customClass e-rte-elements",
        };
    },
    methods: {
        onCreate: function(e) {
        var customBtn = document.getElementById('custom_tbar');
         this.$refs.dialogObj.ej2Instances.target =
document.getElementById('rteSection');
         proxy = this;
        },
        onClick: function() {
            proxy.$refs.customObj.ej2Instances.focusIn();
            proxy.ranges = proxy.selection.getRange(document);
            proxy.$refs.dialogObj.ej2Instances.width =
proxy.$refs.customObj.ej2Instances.element.offsetWidth * 0.5;
            proxy.$refs.dialogObj.ej2Instances.content =
document.getElementById('rteSpecial_char');
            proxy.$refs.dialogObj.ej2Instances.dataBind();
            proxy.$refs.dialogObj.ej2Instances.show();
        }
        dialogCreate: function() {
            var dialogCtn = document.getElementById('rteSpecial_char');
            proxy = this;
            dialogCtn.onclick = function (e) {
            var target = e.target;
            var activeEle =
proxy.$refs.dialogObj.ej2Instances.element.querySelector('.char_block.e-
active');
            if (target.classList.contains('char_block')) {
                target.classList.add('e-active');
                if (activeEle) {
                    activeEle.classList.remove('e-active');
                }
            }
        };
        },
        onInsert: function() {
            var activeEle =
this.$refs.dialogObj.ej2Instances.element.querySelector('.char_block.e-
active');
            if (activeEle) {

proxy.$refs.customObj.ej2Instances.executeCommand('insertText',
activeEle.textContent , {undo: true});
```

```
            }
            this.dialogOverlay();
        },
        dialogOverlay: function() {
             var activeEle =
this.$refs.dialogObj.ej2Instances.element.querySelector('.char_block.e-
active');
            if (activeEle) {
                activeEle.classList.remove('e-active');
            }
            this.$refs.dialogObj.ej2Instances.hide();
        }
    },
    provide:{
        richtexteditor:[Toolbar, Link, Image, QuickToolbar, HtmlEditor]
    }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs19" %}

The focus will be lost while rendering the required component for the custom toolbar, causing it to render outside the Rich Text Editor and triggering a blur event. During that time, proper functionality will not be achievable. Therefore, it is recommended to set the cssClass property or class as e-rte-elements in the dependency component.

Quick inline toolbar

Quick commands are opened as context-menu on clicking the corresponding element. The commands must be passed as string collection to image, text, link and table attributes of the quickToolbarSettings property.

| Target Element | Default Quick Toolbar items |

|----------------|---------|

|image | 'Replace', 'Align', 'Caption', 'Remove', 'InsertLink', 'Display', 'AltText','Dimension'.|

| link | 'Open', 'Edit', 'UnLink'.|

| text | null
 (Any toolbar items in the Rich Text Editor can be configured here).|

Copyright © 2001 -2024 Syncfusion Inc.

| table | 'TableHeader', 'TableRows', 'TableColumns', 'BackgroundColor', 'TableRemove', 'Alignments', 'TableCellVerticalAlign', 'Styles'.|

Custom tool can be added to the corresponding quick toolbar, using quickToolbarSettings property.

The following sample demonstrates the option to insert the image to the Rich Text Editor content as well as option to rotate the image through the quick toolbar. The image rotation functionalities have been achieved through the toolbarClick event.

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section" id="rteSection">
        <ejs-richtexteditor ref="rteObj"
:quickToolbarSettings="quickToolbarSettings" :toolbarClick="onToolbarClick">
        <p>Rich Text Editor allows to insert images from online source as
well as local
            computer where you want to insert the image in your content.</p>
        <p><b>Get started Quick Toolbar to click on the image</b></p>
        <p>It is possible to add custom style on the selected image inside
the Rich Text Editor through quick toolbar.</p>
        <img id="rteImageID" style="width:300px; height:300px;transform:
rotate(0deg);" alt="Logo" src="https://ej2.syncfusion.com/demos/src/rich-
text-editor/images/RTEImage-Feather.png"></ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<style>
.e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
    content: "\e341";
}
.e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
    content: "\e354";
}
</style>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Image,  Link, HtmlEditor,
QuickToolbar, NodeSelection } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
    data: function() {
        return {
        quickToolbarSettings: {
            image: [
                'Replace', 'Align', 'Caption', 'Remove', 'InsertLink',
'OpenImageLink', '-',
                'EditImageLink', 'RemoveImageLink', 'Display', 'AltText',
'Dimension',
                {
                    tooltipText: 'Rotate Left',
```

```
                    template: '<button class="e-tbar-btn e-btn"
id="roatateLeft"><span class="e-btn-icon e-icons e-rotate-left"></span>'
                },
                {
                    tooltipText: 'Rotate Right',
                    template: '<button class="e-tbar-btn e-btn"
id="roatateRight"><span class="e-btn-icon e-icons e-rotate-right"></span>'
                }
            ]
        }
        };
        },
        methods: {
            onToolbarClick: function(e) {
            var nodeObj = new NodeSelection();
            var range =
nodeObj.getRange(this.$refs.rteObj.ej2Instances.contentModule.getDocument())
;
            var imgEle = nodeObj.getNodeCollection(range)[0];
            if (e.item.tooltipText === 'Rotate Right') {
                var transform = (imgEle.style.transform === '') ? 0 :

parseInt(imgEle.style.transform.split('(')[1].split(')')[0], 10);
                imgEle.style.transform = 'rotate(' + (transform + 90) +
'deg)';
            }
            else if (e.item.tooltipText === 'Rotate Left') {
                var transform = (imgEle.style.transform === '') ? 0 :

Math.abs(parseInt(imgEle.style.transform.split('(')[1].split(')')[0], 10));
                imgEle.style.transform = 'rotate(-' + (transform + 90) +
'deg)';
            }
            }
        },
    provide:{
        richtexteditor:[Toolbar, Image,  Link, HtmlEditor, QuickToolbar]
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs20" %}

Copyright © 2001 -2024 Syncfusion Inc.

Rich Text Editor features are segregated into individual feature-wise modules. To use quick toolbar, inject the quick toolbar module using the RichTextEditor.Inject(image, link).

See Also

- How to render the toolbar in inline mode
- How to customize the toolbar items shortcut key

Styling in Vue Rich text editor component

Font name and size

By default, the editor is initialized with font name and font size as **null**. To change it, select a different font name and font size from the drop-down in the editor's toolbar.

To apply different font style for section of the content, select the text that you would like to change, and select a required font style from the drop-down to apply the changes to the selected text.

FontName DropDowns

The following table lists the default font name and width of the fontname drop-down and available list of font names.

| **Property** | **Default value** |

| --- | --- |

| Default font name | null |

| Width | 65px |

| Items | { text: 'Segoe UI', value: 'Segoe UI' },
 { text: 'Arial', value: 'Arial,Helvetica,sans-serif' },
 { text: 'Courier New', value: 'Courier New,Courier,monospace' },
 { text: 'Georgia', value: 'Georgia,serif' },
 { text: 'Impact', value: 'Impact,Charcoal,sans-serif' },
 { text: 'Lucida Console', value: 'Lucida Console,Monaco,monospace' },
 { text: 'Tahoma', value: 'Tahoma,Geneva,sans-serif' },
 { text: 'Times New Roman', value: 'Times New Roman,Times,serif' },
 { text: 'Trebuchet MS', value: 'Trebuchet MS,Helvetica,sans-serif' },
 { text: 'Verdana', value: 'Verdana,Geneva,sans-serif' } |

FontSize DropDowns

The following table list the default font size and width of the fontsize dropdown and available list of font size.

| **Property** | **Default value** |

| --- | --- |

| Default font size | null |

| Width | 35px |

| Items | { text: '8', value: '8pt' },
 { text: '10', value: '10pt' },
 { text: '12', value: '12pt' },
 { text: '14', value: '14pt' },
 { text: '18', value: '18pt' },
 { text: '24', value: '24pt' },
 { text: '36', value: '36pt' } |

The following sample demonstrates the option to add the font name and font size tools to the toolbar as well as modify the default width of the tools.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340"
:toolbarSettings="toolbarSettings" :fontFamily="fontFamily"
:fontSize="fontSize">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        toolbarSettings: {
          items: ['FontName', 'FontSize']
        },
        fontFamily: {
          width: '60px'
        },
        fontSize: {
          width: '40px'
        }
      }
    },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs30" %}

Custom font and size

Rich Text Editor supports to provide custom font and size with existing list.

If you want to add additional font names and font sizes to font drop-down, pass the font information as JSON data to the items field of the fontSize and the fontFamily property.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340"
:toolbarSettings="toolbarSettings" :fontFamily="fontFamily"
:fontSize="fontSize">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
```

```
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        toolbarSettings: {
          items: ['FontName', 'FontSize']
        },
        fontFamily: {
          default: 'Segoe UI',
          width: '60px',
          items: [
            { text: 'Segoe UI', value: 'Segoe UI' },
            { text: 'Arial', value: 'Arial,Helvetica,sans-serif' },
            { text: 'Courier New', value: 'Courier New,Courier,monospace' },
            { text: 'Georgia', value: 'Georgia,serif' },
            { text: 'Impact', value: 'Impact,Charcoal,sans-serif' },
            { text: 'Calibri Light', value: 'CalibriLight' }
          ]
        },
        fontSize: {
          default: '10 pt',
          width: '40px',
          items: [
            { text: '8 pt', value: '8pt' },
            { text: '10 pt', value: '10pt' },
            { text: '12 pt', value: '12pt' },
            { text: '14 pt', value: '14pt' },
            { text: '42 pt', value: '42pt' }
          ]
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
```

```
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-vue-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs31" %}

Font and Background color

To apply font color or background color for a selected content of Rich Text Editor, use font color and background color tools.

Rich Text Editor supports to provide customs font color and background color with existing list through the colorCode field of fontColor and backgroundColor.

The FontColor and the BackgroundColor property has two mode Picker and Palette. Palette mode has predefined set of colorCode and in the picker mode, more colors has been provided. Through modeSwitcher, you can able to switch between these two options.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340"
:toolbarSettings="toolbarSettings" :fontColor="fontColor"
:backgroundColor="backgroundColor">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        toolbarSettings: {
          items: ['FontColor', 'BackgroundColor']
        },
        fontColor: {
          modeSwitcher : true
        },
        backgroundColor: {
          modeSwitcher : true
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs32" %}

Editor content styles

By default, The content styles of Rich Text Editor are not returned while retrieving HTML value from the editor. So, the styles are not applied when using the HTML value outside of the editor. To get the styles

to Rich Text Editor's content for your application, You can copy and use the below styles directly in your application. The styles listed below which used in the UI elements of the Rich Text Editor.

Make sure to add a CSS class 'e-rte-content' to the content container.

```
.e-rte-content p {
margin: 0 0 10px;
margin-bottom: 10px;
}
.e-rte-content li {
margin-bottom: 10px;
}
.e-rte-content h1 {
font-size: 2.17em;
font-weight: 400;
line-height: 1;
margin: 10px 0;
}
.e-rte-content h2 {
font-size: 1.74em;
font-weight: 400;
margin: 10px 0;
}
.e-rte-content h3 {
font-size: 1.31em;
font-weight: 400;
margin: 10px 0;
}
.e-rte-content h4 {
font-size: 1em;
font-weight: 400;
margin: 0;
}
.e-rte-content h5 {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
font-size: 00.8em;
font-weight: 400;
margin: 0;
}
.e-rte-content h6 {
font-size: 00.65em;
font-weight: 400;
margin: 0;
}
.e-rte-content blockquote {
margin: 10px 0;
margin-left: 0;
padding-left: 5px;
}
.e-rte-content pre {
background-color: inherit;
border: 0;
border-radius: 0;
color: #333;
font-size: inherit;
line-height: inherit;
margin: 0 0 10px;
overflow: visible;
padding: 0;
white-space: pre-wrap;
word-break: inherit;
word-wrap: break-word;
}
.e-rte-content strong, .e-rte-content b {
font-weight: 700;
}
.e-rte-content a {
text-decoration: none;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
-webkit-user-select: auto;
-ms-user-select: auto;
user-select: auto;
}
.e-rte-content a:hover {
text-decoration: underline;
}
.e-rte-content h3 + h4,
.e-rte-content h4 + h5,
.e-rte-content h5 + h6 {
margin-top: 00.6em;
}
.e-rte-content .e-rte-image.e-imgbreak {
border: 0;
cursor: pointer;
display: block;
float: none;
margin: 5px auto;
max-width: 100%;
position: relative;
}
.e-rte-content .e-rte-image {
border: 0;
cursor: pointer;
display: block;
float: none;
margin: auto;
max-width: 100%;
position: relative;
}
.e-rte-content .e-rte-image.e-imginline {
display: inline-block;
float: none;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
margin-left: 5px;
margin-right: 5px;
max-width: calc(100% - (2 * 5px));
vertical-align: bottom;
}
.e-rte-content .e-rte-image.e-imgcenter {
cursor: pointer;
display: block;
float: none;
margin: 5px auto;
max-width: 100%;
position: relative;
}
.e-rte-content .e-rte-image.e-imgleft {
float: left;
margin: 0 5px 0 0;
text-align: left;
}
.e-rte-content .e-rte-image.e-imgright {
float: right;
margin: 0 0 0 5px;
text-align: right;
}
.e-rte-content .e-rte-img-caption {
display: inline-block;
margin: 5px auto;
max-width: 100%;
position: relative;
}
.e-rte-content .e-rte-img-caption.e-caption-inline {
display: inline-block;
margin: 5px auto;
margin-left: 5px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
margin-right: 5px;
max-width: calc(100% - (2 * 5px));
position: relative;
text-align: center;
vertical-align: bottom;
}
.e-rte-content .e-rte-img-caption.e-imgcenter {
display: block;
}
.e-rte-content .e-rte-img-caption .e-rte-image.e-imgright,
.e-rte-content .e-rte-img-caption .e-rte-image.e-imgleft {
float: none;
margin: 0;
}
.e-rte-content .e-rte-table {
border-collapse: collapse;
empty-cells: show;
}
.e-rte-content .e-rte-table td,
.e-rte-content .e-rte-table th {
border: 1px solid #bdbdbd;
height: 20px;
min-width: 20px;
padding: 2px 5px;
vertical-align: middle;
}
.e-rte-content .e-rte-table.e-dashed-border td,
.e-rte-content .e-rte-table.e-dashed-border th {
border-style: dashed;
}
.e-rte-content .e-rte-img-caption .e-img-inner {
box-sizing: border-box;
display: block;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
font-size: 16px;
font-weight: initial;
margin: auto;
opacity: .9;
position: relative;
text-align: center;
width: 100%;
}
.e-rte-content .e-rte-img-caption .e-img-wrap {
display: inline-block;
margin: auto;
padding: 0;
width: 100%;
}
.e-rte-content blockquote {
border-left: solid 2px #333;
}
.e-rte-content a {
color: #2e2ef1;
}
.e-rte-content .e-rte-table th {
background-color: #e0e0e0;
}
`
```

See Also

- How to add google fonts to the font family
- How to customize the toolbar items shortcut key
- How to customize the placeholder
- How to change the default font family

Image in Vue Rich text editor component

Rich Text Editor allows to insert images from online sources as well as local computer where you want to insert the image in your content. For inserting the image to the Rich Text Editor, the following list of options have been provided in the insertImageSettings

| Options | Description |

|----------------|---------|

| allowedTypes | Specifies the extensions of the image types allowed to insert on bowering and passing the extensions with comma separators. For example, pass allowedTypes as .jpg and .png.|

| display | Sets the default display for an image when it is inserted in to the Rich Text Editor. Possible options are: 'inline' and 'break'.|

| width | Sets the default width of the image when it is inserted in the Rich Text Editor.|

| height | Sets the default height of the image when it is inserted in the Rich Text Editor.|

| saveUrl | Provides URL to map the action result method to save the image.|

| path | Specifies the location to store the image.|

Upload options

Through the browse option in the Image dialog, select the image from the local machine and insert into the Rich Text Editor content.

If the path field is not specified in the insertImageSettings, the image will be transferred into base 64 and blob url for the image will be created and the generated url will be set to the src property of img tag.

`ts

<img src="blob:http://ej2.syncfusion.com/3ab56a6e-ec0d-490f-85a5-f0aeb0ad8879" >

`

If you want to insert a lot of tiny images in the editor and don't want a specific physical location for saving images, you can opt to save format as Base64.

In the following sample, the image has been loaded from the local machine and it will be saved in the given location.

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj" :height="340"
:toolbarSettings="toolbarSettings"><p>The Rich Text Editor component is
WYSIWYG ("what you see is what you get") editor that provides the best user
experience to create and update the content.
                Users can format their content using standard toolbar
commands.</p>
                <p><b>Key features:</b></p>
                <ul><li><p>Provides < IFRAME > and < DIV > modes</p></li>
                <li><p>Capable of handling markdown editing.</p></li>
                <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
                <li><p>Provides a fully customizable toolbar.</p></li>
                <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
                <li><p>Supports third-party library integration.</p></li>
```

```
                <li><p>Allows preview of modified content before saving
it.</p></li>
                <li><p>Handles images, hyperlinks, video, hyperlinks,
uploads, etc.</p></li>
                </ul></ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor } from
"@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
            toolbarSettings: {
                items: ['Image']
            }
        };
    },
    provide:{
        richtexteditor:[Toolbar, Link, Image, HtmlEditor]
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

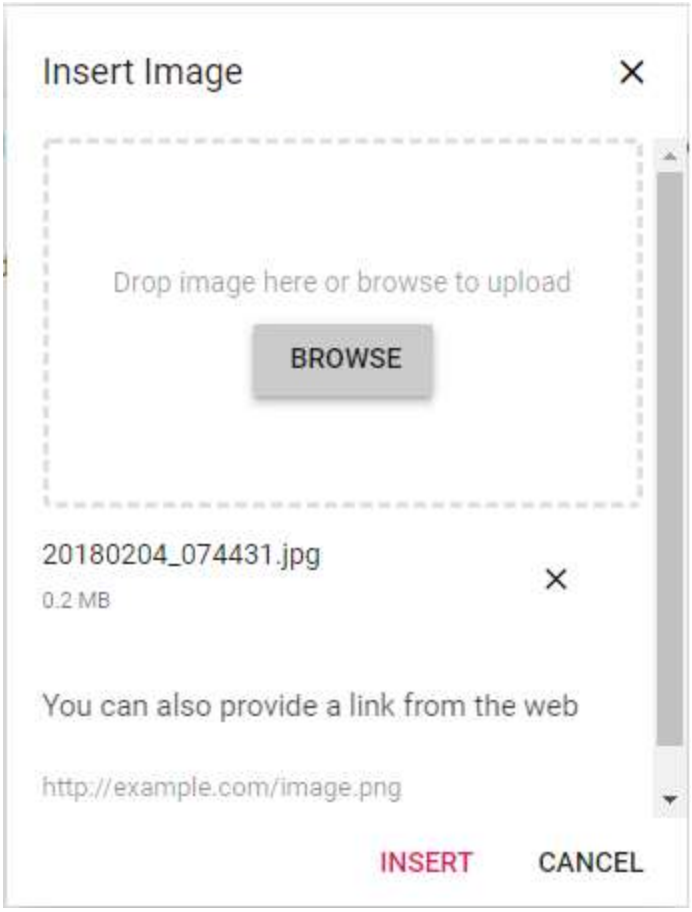
{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs3" %}

Image delete

To remove an image from the Rich Text Editor content, select the image and click Remove tool from the quick toolbar. It will delete the image from the Rich Text Editor content as well as from the service location if the saveUrl is given.

Once you select the image from the local machine, the URL for the image will be generate. From there,you can remove the image from the service location by clicking the cross icon.

Copyright © 2001 -2024 Syncfusion Inc.



The following sample explains, how to configure removeUrl to remove a saved image from the remote service location, when the following image remove actions are performed:

- delete key action.
- backspace key action.
- Removing uploaded image file from the insert image dialog.
- Deleting image using the quick toolbar remove option.

APP.VUE

```
<template>
    <div>
        <div class="control-section">
            <div class="sample-container">
                <div class="default-section">
                    <ejs-richtexteditor :toolbarSettings="toolbarSettings"
:insertImageSettings="insertImageSettings">
                        <p>The Rich Text Editor component is WYSIWYG ("what
you see is what you get") editor that provides the best user experience to
create and update the content. Users can format their content using standard
toolbar commands.</p>
                        <p><b>Key features:</b></p>
                        <ul>
```

```
                            <li><p>Provides < IFRAME > and < DIV >
modes</p></li>
                            <li><p>Capable of handling markdown
editing.</p></li>
                            <li><p>Contains a modular library to load the
necessary functionality on demand.</p></li>
                            <li><p>Provides a fully customizable
toolbar.</p></li>
                            <li><p>Provides HTML view to edit the source
directly for developers.</p></li>
                            <li><p>Supports third-party library
integration.</p></li>
                            <li><p>Allows preview of modified content before
saving it.</p></li>
                            <li><p>Handles images, hyperlinks, video,
hyperlinks, uploads, etc.</p></li>
                        </ul>
                    </ejs-richtexteditor>
                </div>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor } from
"@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
            toolbarSettings: {
                items: ['Image']
            },
            insertImageSettings: {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            }
        };
    },
    provide:{
        richtexteditor:[Toolbar, Link, Image, HtmlEditor]
    }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
```

```
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs18" %}

Insert from web

To insert an image from the online source like Google, Ping, etc., you should enable the image tool on the editor's toolbar. By default, the image tool opens a simple dialog which allows you to insert an image from online source.

Dimension

Sets the default width and height of the image when it is inserted in the Rich Text Editor using width and height of the insertImageSettings.

Through the quickToolbar, also you can change the width and height using Change Size option. Once click into the option. The Image Size dialog will open as below. In that specify the width and height of the image in pixel.

![image-size](./images/image-size.png)

Caption and Alt Text

Image caption and alternative text can be specified for the inserted image in the Rich Text Editor through the quick toolbar options such as, Image Caption and Alternative Text.

Through the Alternative Text option, set the alternative text for the image, when the image is not upload successfully into the Rich Text Editor.

By clicking the Image Caption, the image will get wrapped in an image element with a caption. Then, you can type caption content inside the Rich Text Editor.

Display position

Sets the default display for an image when it is inserted in the Rich Text Editor using display field in insertImageSettings. It has two possible options: 'inline' and 'break'.

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj"
:insertImageSettings="insertImageSettings"><p>Rich Text Editor allows to
insert images from online source as well as local
            computer where you want to insert the image in your content.</p>
            <p><b>Get started Quick Toolbar to click on the image</b></p>
            <p>It is possible to add custom style on the selected image
inside the Rich Text Editor through quick toolbar.</p>
            <img id="rteImageID" style="width:300px; height:300px;transform:
rotate(0deg);" alt="Logo" src="https://ej2.syncfusion.com/demos/src/rich-
text-editor/images/RTEImage-Feather.png"></ejs-richtexteditor>
        </div>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

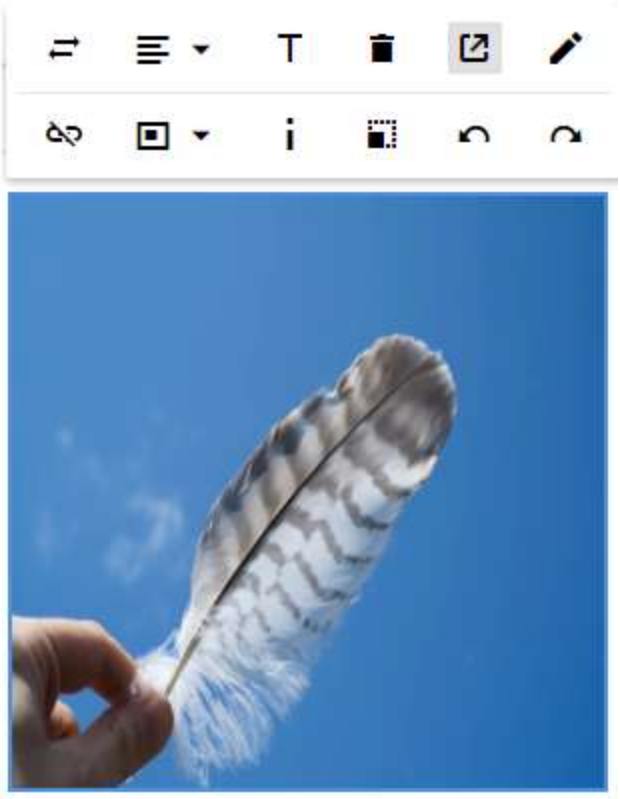
```
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Image,  Link, HtmlEditor,
QuickToolbar, NodeSelection } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
    data: function() {
        return {
            insertImageSettings: {
                display: 'inline'
            }
        };
    },
    provide:{
        richtexteditor:[Toolbar, Image,  Link, HtmlEditor, QuickToolbar]
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs4" %}

Image with link

The hyperlink itself can be an image in Rich Text Editor. If the image given as hyperlink, the remove, edit, and open link will be added to the quick toolbar of image. For further details about link, see the link documentation documentation.

Copyright © 2001 -2024 Syncfusion Inc.



Resize

Rich Text Editor has a built-in image inserting support. The resize points will be appearing on each corner of image when focus. So, users can resize the image using mouse points or thumb through the resize points easily. Also, the resize calculation will be done based on aspect ratio.

![image-resize](./images/image-resize.png)

Drag and Drop

By default, the Rich Text Editor allows you to insert images by drag-and-drop from the local file system such as Windows Explorer into the content editor area. And, you can upload the images to the server before inserting into the editor by configuring the saveUrl property. The images can be repositioned anywhere within the editor area by dragging and dropping the image.

In the following sample, you can see feature demo.

APP.VUE

```
 html
<template>
    <div>
        <div class="control-section">
            <div class="sample-container">
                <div class="default-section">
                    <ejs-richtexteditor ref="rteObj"
:insertImageSettings="insertImageSettings">
                        <p>The Rich Text Editor component is WYSIWYG ("what
you see is what you get") editor that provides the best user experience to
create and update the content.Users can format their content using standard
toolbar commands.</p>
```

```
                    <p><b>Key features:</b></p>
                    <ul>
                        <li><p>Provides IFRAME and DIV modes</p></li>
                        <li><p>Capable of handling markdown
editing.</p></li>
                        <li><p>Contains a modular library to load the
necessary functionality on demand.</p></li>
                        <li><p>Provides a fully customizable
toolbar.</p></li>
                        <li><p>Provides HTML view to edit the source
directly for developers.</p></li>
                        <li><p>Supports third-party library
integration.</p></li>
                        <li><p>Allows preview of modified content before
saving it.</p></li>
                        <li><p>Handles images, hyperlinks, video,
hyperlinks, uploads, etc.</p></li>
                        <li><p>Contains undo/redo manager.</p></li>
                        <li><p>Creates bulleted and numbered lists.</p></li>
                    </ul>
                    </ejs-richtexteditor>
                </div>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import {
        RichTextEditorPlugin,
        Toolbar,
        Image,
        Link,
        HtmlEditor,
        QuickToolbar
    } from "@syncfusion/ej2-vue-richtexteditor";
    Vue.use(RichTextEditorPlugin);
    export default {
        data: function() {
            return {
                insertImageSettings: {
                    saveUrl :
'https://aspnetmvc.syncfusion.com/services/api/uploadbox/Save'
                }
            };
        },
        provide: {
            richtexteditor: [Toolbar, Image, Link, HtmlEditor, QuickToolbar]
        }
    }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs19" %}

Drag and drop with specific extension images

You can allow the specific images alone to be uploaded using the the allowedExtentions property. The extension can be represented as collection by comma separators. By default, the Rich Text Editor allows the JPG, JPEG, and PNG formats. You can configure this formats as follows.

`ts

insertImageSettings: {

allowedTypes: ['.jpg']

}

`

Prevent drag and drop action

You can prevent drag-and-drop action by setting the actionBegin argument cancel value to true. The following code shows how to prevent the drag-and-drop.

`ts

actionBegin: function (args: any): void {

if(args.type === 'drop' || args.tpe === 'dragstart') {

args.cancel =true;

}

}

`

See Also

- How to edit the quick toolbar settings
- How to use link editing option in the toolbar items

Audio in Vue Rich text editor component

The Rich Text Editor allows you to insert audio from online sources and local computers and then insert them into your content. You can insert the audio with the following list of options in the insertAudioSettings property.

| Options | Description |

|----------------|---------|

Copyright © 2001 -2024 Syncfusion Inc.

| allowedTypes | Specifies the extensions of the audio types allowed to insert on bowering and passing the extensions with comma separators. For example, pass allowedTypes as .mp3, .wav, .m4a and .wma. |

| layoutOption | Sets the default display for audio when it is inserted into the Rich Text Editor. Possible options are Inline and Break.|

| saveFormat | Sets the default save format of the audio element when inserted. Possible options are: Blob and Base64.|

| saveUrl | Provides URL to map the action result method to save the audio.|

| removeUrl | Provides URL to map the action result method to remove the audio.|

| path | Specifies the location to store the audio.|

Configure audio tool in the toolbar

You can add an audio tool in the Rich Text Editor toolbar using the toolbarSettings items property.

Rich Text Editor features are segregated into individual feature-wise modules. To use audio, inject the Audio module in provides section.

To configure the Audio toolbar item, refer to the below code.

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj" :height = "340"
:insertAudioSettings = "insertAudioSettings" :toolbarSettings =
"toolbarSettings" >
       <p>The Rich Text Editor component is WYSIWYG ("what you see is what
you get") editor that provides the best user experience to create and update
the content.
          Users can format their content using standard toolbar
commands.</p>
            <p><b>Key features:</b></p>
            <ul>
            <li><p>Provides < IFRAME > and < DIV > modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </ejs-richtexteditor>
        </div>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Audio, HtmlEditor,
QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
            toolbarSettings: {
                items: ['Audio']
            },
            insertAudioSettings: {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            }
        };
    },
    provide:{
        richtexteditor:[Toolbar, Link, Audio, HtmlEditor, QuickToolbar]
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import '../node_modules/@syncfusion/ej2-
richtexteditor/styles/material.css';
@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-gantt/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/audio-cs1" %}

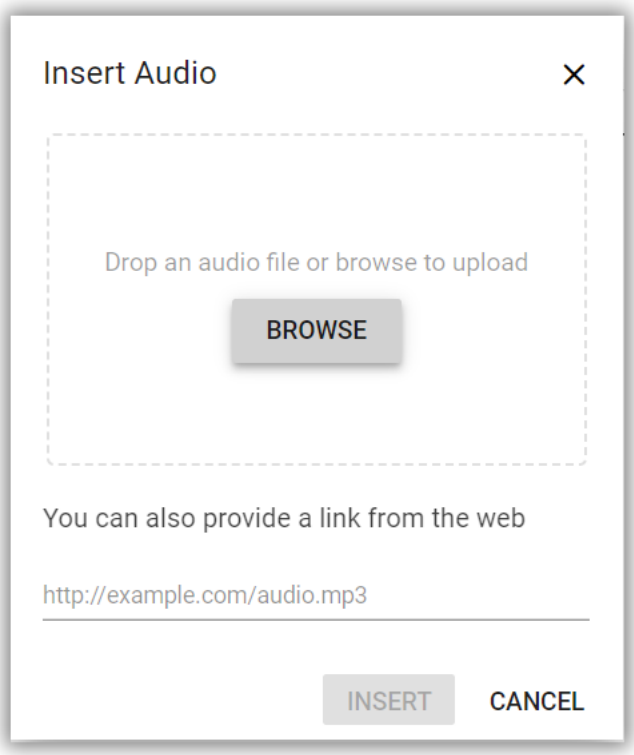
Insert audio from the web

You can insert audio from either the hosted link or the local machine, by clicking the audio button in the editor's toolbar. On clicking the audio button, a dialog opens, which allows you to insert audio from the web URL.

Insert from web URL

By default, the audio tool opens the audio dialog, allowing you to insert audio from an online source. Inserting the URL will be added to the `src` attribute of the `<source>` tag.

Copyright © 2001 -2024 Syncfusion Inc.



Insert audio from local machine

You can use the browse option on the audio dialog, to select the audio from the local machine and insert it into the Rich Text Editor content.

If the path field is not specified in the insertAudioSettings, the audio will be converted into the Blob URL or Base64 and inserted inside the Rich Text Editor.

Restrict audio upload based on size

You can restrict the audio uploaded from the local machine when the uploaded audio file size is greater than the allowed size by using the fileUploading event.

The file size in the argument will be returned in bytes.

In the following illustration, the audio size has been validated before uploading, and it is determined whether the audio has been uploaded or not.

`

<template>

<ejs-richtexteditor ref="rteObj" :height = "340" :insertAudioSettings = "insertAudioSettings" :toolbarSettings = "toolbarSettings" :fileUploading = "onFileUpload">

</ejs-richtexteditor>

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Audio, HtmlEditor, QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
data: function() {
return {
toolbarSettings: {
items: ['Audio']
},
insertAudioSettings: {
saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
removeUrl:  'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
}
};
},
methods: {
onFileUpload: function(args) {
let sizeInBytes = args.fileData.size;
let fileSize = 500000;
if (fileSize < sizeInBytes) {
args.cancel = true;
}
}
},
provide:{
richtexteditor:[Toolbar, Link, Audio, HtmlEditor, QuickToolbar]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
}
</script>
<style>
@import 'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import '../node_modules/@syncfusion/ej2-richtexteditor/styles/material.css';
@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-gantt/styles/material.css";
</style>
`
```

Server-side action

The selected audio can be uploaded to the required destination using the controller action below. Map this method name in insertAudioSettings.saveUrl and provide the required destination path through insertAudioSettings.path properties.

If you want to insert lower-sized audio files in the editor and don't want a specific physical location for saving the audio, you can opt to save the format as `Base64`.

In the following code blocks, the audio module has been injected and can insert the audio files saved in the specified path.

```
`
<template>
<div>
<div class="control-section">
<div class="sample-container">
<div class="default-section">
```

```
<ejs-richtexteditor ref="rteObj" :height="340" :toolbarSettings="toolbarSettings" :insertAudioSettings =
"insertAudioSettings" >
</ejs-richtexteditor>
</div>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Audio, HtmlEditor, QuickToolbar } from "@syncfusion/ej2-
vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
data: function() {
return {
toolbarSettings: {
items: ['Audio']
},
insertAudioSettings: {
saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/SaveFiles",
path: "[SERVICEHOSTEDPATH]/Files/"
}
};
},
provide:{
richtexteditor:[Toolbar, Link, Audio, HtmlEditor, QuickToolbar]
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
```

```
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import '../node_modules/@syncfusion/ej2-richtexteditor/styles/material.css';
@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-gantt/styles/material.css";
</style>
`
```

```c#
using System;
using System.IO;
using FileUpload.Models;
using System.Diagnostics;
using System.Net.Http.Headers;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using System.Collections.Generic;
using Microsoft.AspNetCore.Hosting;
namespace FileUpload.Controllers
{
public class HomeController : Controller
{
private IHostingEnvironment hostingEnv;
public HomeController(IHostingEnvironment env)
{
hostingEnv = env;
}
public IActionResult Index()
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{
return View();
}
[AcceptVerbs("Post")]
public void SaveFiles(IList<IFormFile> UploadFiles)
{
try
{
foreach (IFormFile file in UploadFiles)
{
if (UploadFiles != null)
{
string filename = ContentDispositionHeaderValue.Parse(file.ContentDisposition).FileName.Trim('"');
filename = hostingEnv.WebRootPath + "\\Files" + $@"\{filename}";
// Create a new directory, if it does not exists
if (!Directory.Exists(hostingEnv.WebRootPath + "\\Files"))
{
Directory.CreateDirectory(hostingEnv.WebRootPath + "\\Files");
}
if (!System.IO.File.Exists(filename))
{
using (FileStream fs = System.IO.File.Create(filename))
{
file.CopyTo(fs);
fs.Flush();
}
Response.StatusCode = 200;
}
}
}
}
catch (Exception)
{
```

```
Response.StatusCode = 204;
}
}
[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
public IActionResult Error()
{
return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
}
}
}
```

Audio save format

The audio files can be saved as `Blob` or `Base64` URL by using the insertAudioSettings.saveFormat property, which is of enum type, and the generated URL will be set to the `src` attribute of the `<source>` tag.

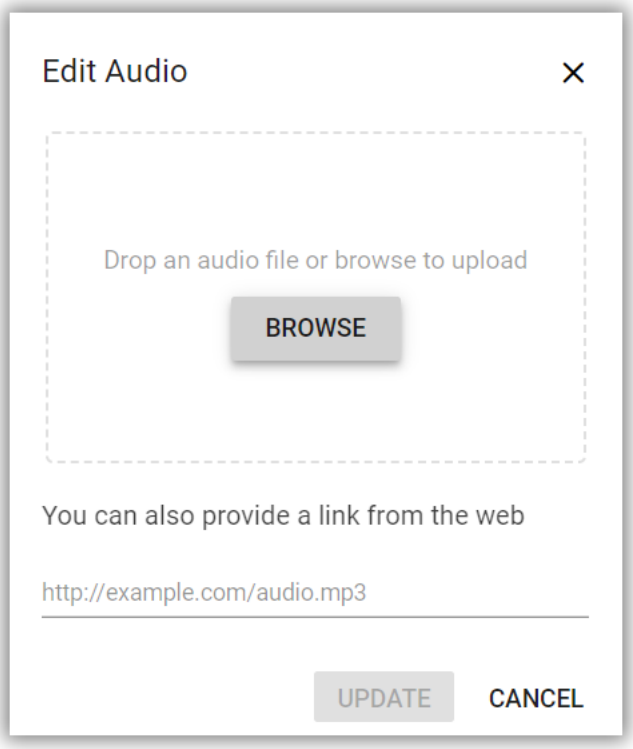
The default `saveFormat` property is set to `Blob` format.

```html
<audio>
<source src="blob:http://ej2.syncfusion.com/3ab56a6e-ec0d-490f-85a5-f0aeb0ad8879" type="audio/mp3" >
</audio>
<audio>
<source src="data:audio/mp3base64,iVBORw0KGgoAAAANSUhEUgAAADAAAAAwCAYAAABXAvmHA" type="audio/mp3" >
</audio>
```

Replacing audio

Once an audio file has been inserted, you can change it using the Rich Text Editor quickToolbarSettings `audioReplace` option. You can replace the audio file using the web URL or the browse option in the audio dialog.

Copyright © 2001 -2024 Syncfusion Inc.

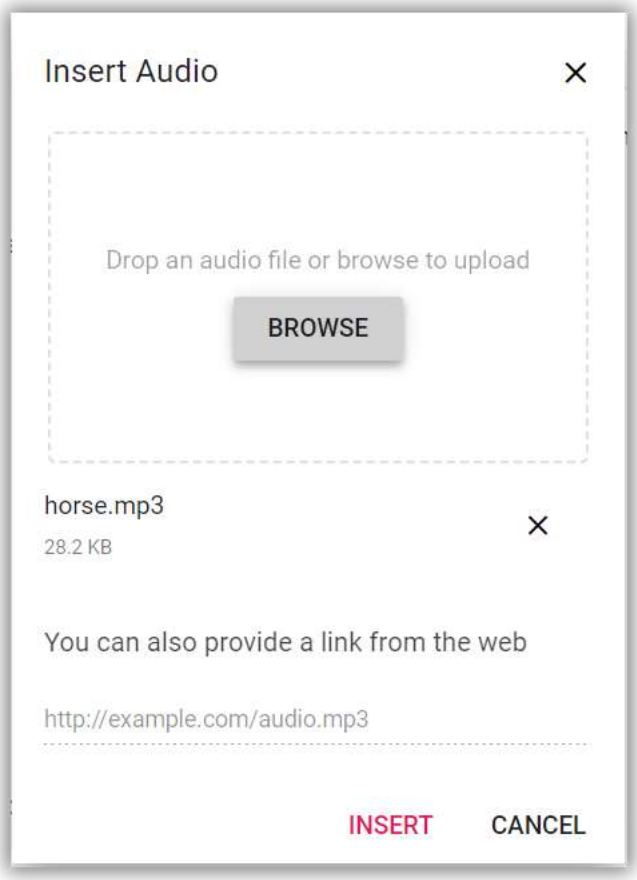


Delete audio

To remove audio from the Rich Text Editor content, select the audio and click the audioRemove button from the quick toolbar. It will delete the audio from the Rich Text Editor content as well as from the service location if the insertAudioSettings.removeUrl is given.

Once you select the audio from the local machine, the URL for the audio will be generated. You can remove the audio from the service location by clicking the cross icon.

Copyright © 2001 -2024 Syncfusion Inc.



Display position

Sets the default display property for audio when it is inserted in the Rich Text Editor using the insertAudioSettings.layoutOption property. It has two possible options: Inline and Break. When updating the display positions, it updates the audio elements' layout position.

The default layoutOption property is set to Inline.

`

```
<template>
<div>
<div class="control-section">
<div class="sample-container">
<div class="default-section">
<ejs-richtexteditor ref="rteObj" :height = "340" :insertAudioSettings = "insertAudioSettings"
:toolbarSettings = "toolbarSettings" >
</ejs-richtexteditor>
</div>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Audio, HtmlEditor, QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
data: function() {
return {
toolbarSettings: {
items: ['Audio']
},
insertAudioSettings: {
layoutOption: 'Inline'
}
};
},
provide:{
richtexteditor:[Toolbar, Link, Audio, HtmlEditor, QuickToolbar]
}
}
</script>
<style>
@import 'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import '../node_modules/@syncfusion/ej2-richtexteditor/styles/material.css';
@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-gantt/styles/material.css";
</style>
`
```

Rename audio before inserting

You can use the insertAudioSettings property, to specify the server handler to upload the selected audio. Then by binding the fileUploadSuccess event, you can receive the modified file name from the server and update it in the Rich Text Editor's insert audio dialog.

```
`
<template>
<div>
<div class="control-section">
<div class="sample-container">
<div class="default-section">
<ejs-richtexteditor ref="rteObj" :height = "340" :insertAudioSettings = "insertAudioSettings" :toolbarSettings = "toolbarSettings" :fileUploadSuccess = "onFileUploadSuccess">
<p>The Rich Text Editor is WYSIWYG ("what you see is what you get") editor useful to create and edit content, and return the valid <a href="https://ej2.syncfusion.com/home/" target="blank”>HTML markup</a> or <a href=“https://ej2.syncfusion.com/home/” target=“blank">markdown</a> of the content</p>
</ejs-richtexteditor>
</div>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Audio, HtmlEditor, QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
```

```
data: function() {
return {
toolbarSettings: {
items: ['Audio']
},
insertAudioSettings: {
saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/Rename",
path: "[SERVICEHOSTEDPATH]/Files/"
}
};
},
methods: {
onFileUploadSuccess (args) {
alert("Get the new file name here");
if (args.e.currentTarget.getResponseHeader('name') != null) {
args.file.name = args.e.currentTarget.getResponseHeader('name');
let filename = document.querySelectorAll(".e-file-name")[0];
filename.innerHTML = args.file.name.replace(document.querySelectorAll(".e-file-type")[0].innerHTML,
'');
filename.title = args.file.name;
}
}
},
provide:{
richtexteditor:[Toolbar, Link, Audio, HtmlEditor, QuickToolbar]
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import '../node_modules/@syncfusion/ej2-richtexteditor/styles/material.css';
@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-gantt/styles/material.css";
</style>
`
```

To configure the server-side handler, refer to the below code.

```c#
int x = 0;
string file;
[AcceptVerbs("Post")]
public void Rename()
{
try
{
var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
fileName = httpPostedFile.FileName;
if (httpPostedFile != null)
{
var fileSave = System.Web.HttpContext.Current.Server.MapPath("~/Files");
if (!Directory.Exists(fileSave))
{
Directory.CreateDirectory(fileSave);
}
var fileName = Path.GetFileName(httpPostedFile.FileName);
var fileSavePath = Path.Combine(fileSave, fileName);
while (System.IO.File.Exists(fileSavePath))
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{
fileName = "rteFiles" + x + "-" + fileName;
fileSavePath = Path.Combine(fileSave, fileName);
x++;
}
if (!System.IO.File.Exists(fileSavePath))
{
httpPostedFile.SaveAs(fileSavePath);
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.Headers.Add("name", fileName);
Response.ContentType = "application/json; charset=utf-8";
Response.StatusDescription = "File uploaded succesfully";
Response.End();
}
}
}
catch (Exception e)
{
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusCode = 204;
Response.Status = "204 No Content";
Response.StatusDescription = e.Message;
Response.End();
}
}
`
```

Upload audio with authentication

You can add additional data with the audio uploaded from the Rich Text Editor on the client side, which can even be received on the server side by using the fileUploading event and its customFormData argument, you can pass parameters to the controller action. On the server side, you can fetch the custom headers by accessing the form collection from the current request, which retrieves the values sent using the POST method.

By default, it doesn't support the UseDefaultCredentials property; we need to manually append the default credentials with the upload request.

```
<template>
<div>
<div class="control-section">
<div class="sample-container">
<div class="default-section">
<ejs-richtexteditor ref="rteObj" :height = "340" :insertAudioSettings = "insertAudioSettings" :toolbarSettings = "toolbarSettings" :fileUploading = "onFileUpload">
</ejs-richtexteditor>
</div>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Audio, HtmlEditor, QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
data: function() {
return {
toolbarSettings: {
items: ['Audio']
},
insertAudioSettings: {
saveUrl: "[SERVICE*HOSTED*PATH]/api/uploadbox/SaveFiles",
path: "[SERVICE*HOSTED*PATH]/Files/"
}
};
},
methods: {
```

```
onFileUpload (args) {
alert("Get the new file name here");
var accessToken = "Authorization_token";
// adding custom Form Data
args.customFormData = [{ 'Authorization': accessToken }];
}
},
provide:{
richtexteditor:[Toolbar, Link, Audio, HtmlEditor, QuickToolbar]
}
}
</script>
<style >
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import '../node_modules/@syncfusion/ej2-richtexteditor/styles/material.css';
@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-gantt/styles/material.css";
</style>
`
`c#
public void SaveFiles(IList<IFormFile> UploadFiles)
{
string currentPath = Request.Form["Authorization"].ToString();
```

Copyright © 2001 -2024 Syncfusion Inc.

}
`

See Also

- How to edit the quick toolbar settings
- How to use the link editing option in the toolbar items

Video in Vue Rich text editor component

The Rich Text Editor allows you to insert videos from online sources and local computers and then insert them into your content. You can insert the video with the following list of options in the insertVideoSettings property.

| Options | Description |

|----------------|---------|

| allowedTypes | Specifies the extensions of the video types allowed to insert on bowering and passing the extensions with comma separators. For example, pass allowedTypes as .mp4, .mov, .wmv and .avi.|

| layoutOption | Sets the default display for a video when it is inserted into the Rich Text Editor. Possible options are: Inline and Break.|

| saveFormat | Sets the default save format of the video element when inserted. Possible options are: Blob and Base64.|

| width | Sets the default width of the video when it is inserted in the Rich Text Editor.|

| minWidth | Sets the minWidth of the video element when it is inserted in the Rich Text Editor.|

| maxWidth | Sets the maxWidth of the video element when it is inserted in the Rich Text Editor.|

| height | Sets the default height of the video when it is inserted in the Rich Text Editor.|

| minHeight | Sets the minHeight of the video element when it is inserted in the Rich Text Editor.|

| maxHeight | Sets the maxHeight of the video element when it is inserted in the Rich Text Editor.|

| saveUrl | Provides URL to map the action result method to save the video.|

| removeUrl | Provides URL to map the action result method to remove the video.|

| path | Specifies the location to store the video.|

| resize | Sets the resizing action for the video element.|

| resizeByPercent | Sets the percentage values for the video element with the resizing action.|

Configure the video tool in the toolbar

You can add the video tool in the Rich Text Editor toolbar using the toolbarSettings items property.

Rich Text Editor features are segregated into individual feature-wise modules. To use the video, inject the Video module in provides section.

To configure the Video toolbar item, refer to the below code.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj" :height = "340"
:insertVideoSettings = "insertVideoSettings" :toolbarSettings =
"toolbarSettings" >
             <p>The Rich Text Editor component is WYSIWYG ("what you see is
what you get") editor that provides the best user experience to create and
update the content.
          Users can format their content using standard toolbar
commands.</p>
            <p><b>Key features:</b></p>
            <ul>
            <li><p>Provides < IFRAME > and < DIV > modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Video, HtmlEditor,
QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
            toolbarSettings: {
                items: ['Video']
            },
            insertVideoSettings: {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            }
        };
    },
    provide:{
        richtexteditor:[Toolbar, Link, Video, HtmlEditor, QuickToolbar]
    }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import '../node_modules/@syncfusion/ej2-
richtexteditor/styles/material.css';
@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-gantt/styles/material.css";
</style>
```

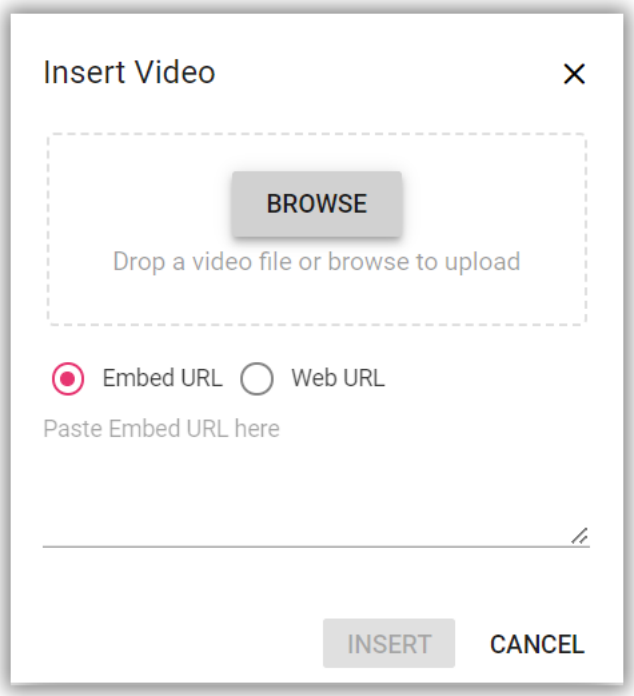
{% previewsample "page.domainurl/code-snippet/rich-text-editor/video-cs1" %}

Insert a video from the web

You can insert a video from either the hosted link or the local machine by clicking the video button in the editor's toolbar. On Clicking the Video button, a dialog opens which allows you to insert video from the Embedded URL or web URL.

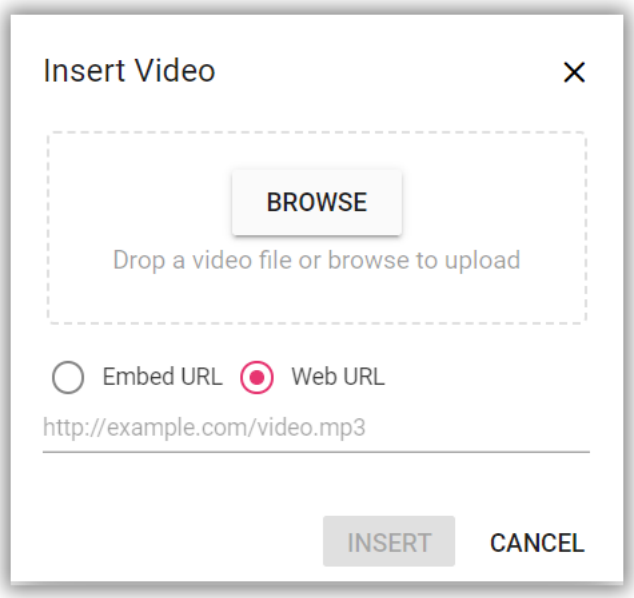
Insert from embed URL

The insert video dialog opens with the Embed URL option as default which allows you to insert an embedded URL.



Insert from web URL

You can switch to Web URL by selecting the web URL check box. Inserting with the web URL option will add the video URL as the src attribute of the <source> tag.



Insert video from local machine

You can use the browse option on the video dialog, to select the video from the local machine and insert it into the Rich Text Editor content.

If the path field is not specified in the insertVideoSettings, the video will be converted into the Blob URL or Base64 and inserted inside the Rich Text Editor.

Restrict video upload based on size

You can restrict the video uploaded from the local machine when the uploaded video file size is greater than the allowed size by using the fileUploading event.

The file size in the argument will be returned in bytes.

In the following example, the video size has been validated before uploading and determined whether the video has been uploaded or not.

`

<template>

<ejs-richtexteditor ref="rteObj" :height = "340" :insertVideoSettings = "insertVideoSettings" :toolbarSettings = "toolbarSettings" :fileUploading = "onFileUpload">

</ejs-richtexteditor>

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Video, HtmlEditor, QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
data: function() {
return {
toolbarSettings: {
items: ['Video']
},
insertVideoSettings: {
saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
removeUrl:  'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
}
};
},
methods: {
onFileUpload: function(args) {
let sizeInBytes = args.fileData.size;
let fileSize = 500000;
if (fileSize < sizeInBytes) {
args.cancel = true;
}
}
},
provide:{
richtexteditor:[Toolbar, Link, Video, HtmlEditor, QuickToolbar]
```

Copyright © 2001 -2024 Syncfusion Inc.

}

}

</script>

<style>

@import 'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-base/styles/material.css';

@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';

@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';

@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';

@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';

@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';

@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';

@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';

@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';

@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';

@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';

@import '../node_modules/@syncfusion/ej2-richtexteditor/styles/material.css';

@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';

@import "../node_modules/@syncfusion/ej2-vue-gantt/styles/material.css";

</style>

`

Server-side action

The selected video can be uploaded to the required destination using the controller action below. Map this method name in insertVideoSettings.saveUrl and provide the required destination path through insertVideoSettings.path properties.

If you want to insert lower-sized video files in the editor and don't want a specific physical location for saving the video, you can save the format as `Base64`.

In the following code blocks, the video module has been injected and can insert the video files saved in the specified path.

`ts

<template>

```
<ejs-richtexteditor ref="rteObj" :height="340" :toolbarSettings="toolbarSettings" :insertVideoSettings =
"insertVideoSettings" >
</ejs-richtexteditor>
</div>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Video, HtmlEditor, QuickToolbar } from "@syncfusion/ej2-
vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
data: function() {
return {
toolbarSettings: {
items: ['Video']
},
insertVideoSettings: {
saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/SaveFiles",
path: "[SERVICEHOSTEDPATH]/Files/"
}
};
},
provide:{
richtexteditor:[Toolbar, Link, Video, HtmlEditor, QuickToolbar]
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import '../node_modules/@syncfusion/ej2-richtexteditor/styles/material.css';
@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-gantt/styles/material.css";
</style>
`
```

```c#
using System;
using System.IO;
using FileUpload.Models;
using System.Diagnostics;
using System.Net.Http.Headers;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using System.Collections.Generic;
using Microsoft.AspNetCore.Hosting;
namespace FileUpload.Controllers
{
public class HomeController : Controller
{
private IHostingEnvironment hostingEnv;
public HomeController(IHostingEnvironment env)
{
hostingEnv = env;
}
public IActionResult Index()
```

```
{
return View();
}
[AcceptVerbs("Post")]
public void SaveFiles(IList<IFormFile> UploadFiles)
{
try
{
foreach (IFormFile file in UploadFiles)
{
if (UploadFiles != null)
{
string filename = ContentDispositionHeaderValue.Parse(file.ContentDisposition).FileName.Trim('"');
filename = hostingEnv.WebRootPath + "\\Files" + $@"\{filename}";
// Create a new directory, if it does not exists
if (!Directory.Exists(hostingEnv.WebRootPath + "\\Files"))
{
Directory.CreateDirectory(hostingEnv.WebRootPath + "\\Files");
}
if (!System.IO.File.Exists(filename))
{
using (FileStream fs = System.IO.File.Create(filename))
{
file.CopyTo(fs);
fs.Flush();
}
Response.StatusCode = 200;
}
}
}
}
catch (Exception)
{
```

```
Response.StatusCode = 204;
}
}
[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
public IActionResult Error()
{
return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
}
}
}
```

Video save format

The video files can be saved as `Blob` or `Base64` URL by using the insertVideoSettings.saveFormat property, which is of enum type, and the generated URL will be set to the `src` attribute of the `<source>` tag.

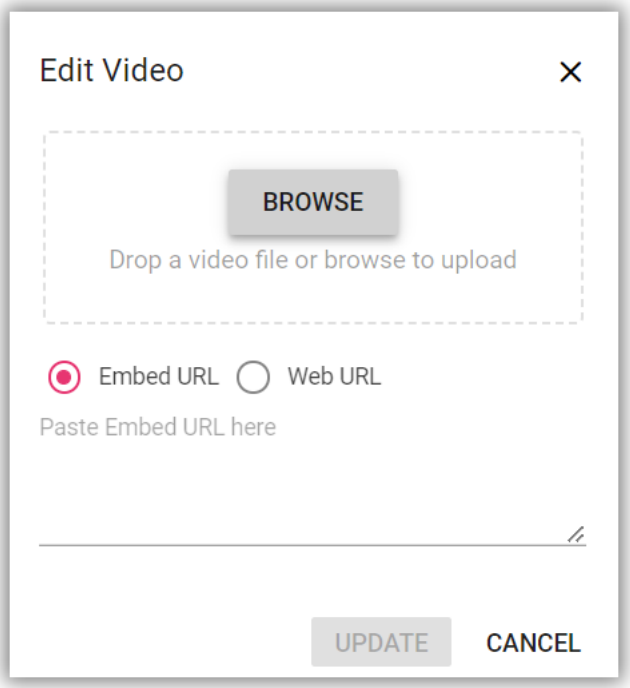
The default `saveFormat` property is set to `Blob` format.

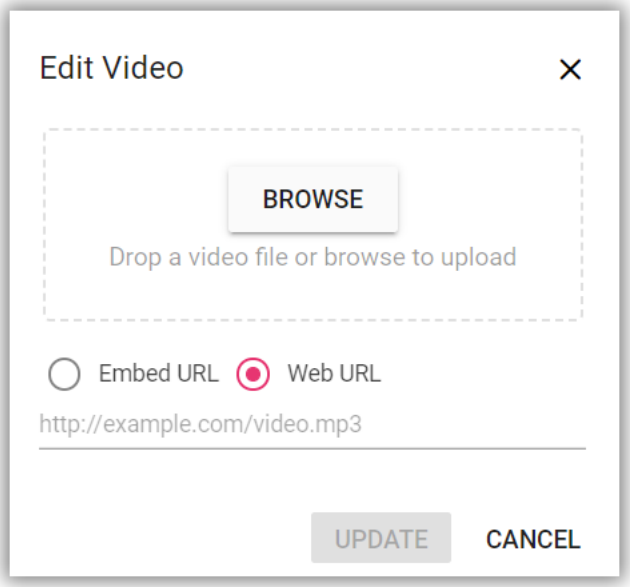
```html
<video>
<source src="blob:http://ej2.syncfusion.com/3ab56a6e-ec0d-490f-85a5-f0aeb0ad8879" type="video/mp4" >
</video>
<video>
<source src="data:video/mp4base64,iVBORw0KGgoAAAANSUhEUgAAADAAAAAwCAYAAABXAvmHA" type="video/mp4" >
</video>
```

Replacing video

Once a video file has been inserted, you can replace it using the Rich Text Editor quickToolbarSettings `videoReplace` option. You can replace the video file either by using the embedded URL or the web URL and the browse option in the video dialog.

Copyright © 2001 -2024 Syncfusion Inc.



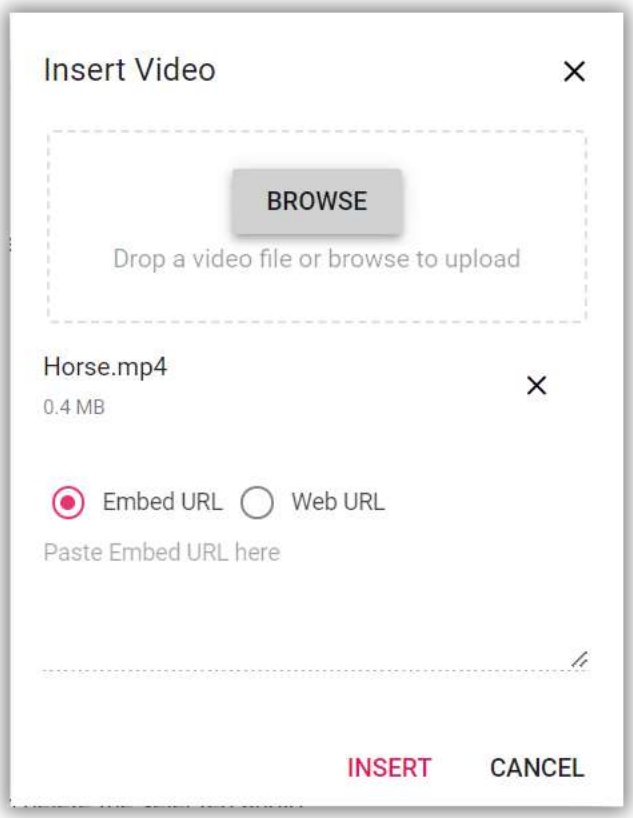


Delete video

To remove a video from the Rich Text Editor content, select the video and click the videoRemove button from the quick toolbar. It will delete the video from the Rich Text Editor content as well as from the service location if the insertVideoSettings.removeUrl is given.

Once you select the video from the local machine, the URL for the video will be generated. You can remove the video from the service location by clicking the cross icon.

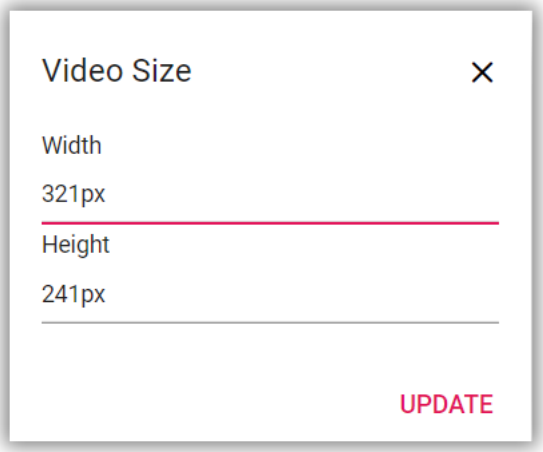
Copyright © 2001 -2024 Syncfusion Inc.



Dimension

Set the default width, minWidth, height, and minHeight of the video element, when it is inserted in the Rich Text Editor using the width, minWidth, height, minHeight properties.

Through the quickToolbarSettings, also you can change the width and height using the Change Size button. Once you click on the button, the video size dialog will open as below. In that, specify the width and height of the video in pixels.



Copyright © 2001 -2024 Syncfusion Inc.

Display position

Sets the default display property for the video when it is inserted in the Rich Text Editor using the insertVideoSettings.layoutOption property. It has two possible options: Inline and Break. When updating the display positions, it updates the video elements' layout position.

The default layoutOption property is set to Inline.

```
<template>
<div>
<div class="control-section">
<div class="sample-container">
<div class="default-section">
<ejs-richtexteditor ref="rteObj" :height = "340" :insertVideoSettings = "insertVideoSettings" :toolbarSettings = "toolbarSettings" >
</ejs-richtexteditor>
</div>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Video, HtmlEditor, QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
data: function() {
return {
toolbarSettings: {
items: ['Video']
},
insertVideoSettings: {
layoutOption: 'Inline'
}
};
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
provide:{
richtexteditor:[Toolbar, Link, Video, HtmlEditor, QuickToolbar]
}
}
</script>
<style>
@import 'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import '../node_modules/@syncfusion/ej2-richtexteditor/styles/material.css';
@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-gantt/styles/material.css";
</style>
`
```

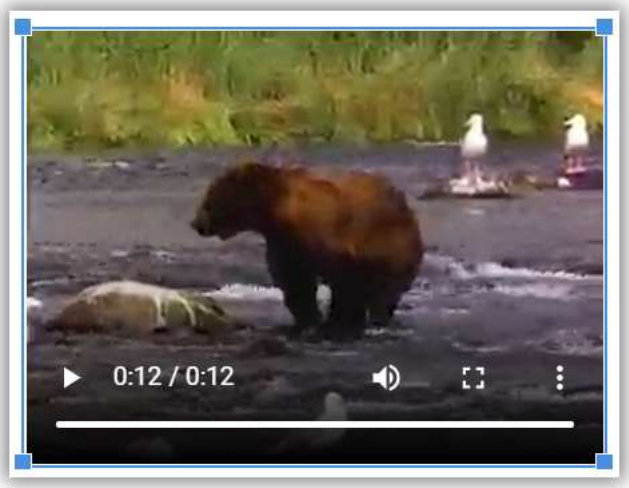
Resize video

The Rich Text Editor has built-in video resizing support, which is enabled for the video elements added. The resize points will appear on each corner of the video when focusing so users can easily resize the video using mouse points or thumb through the resize points. Also, the resize calculation will be done based on the aspect ratio.

You can disable the resize action by configuring `false` for the insertVideoSettings.resize property.

If the minWidth and minHeight properties are configured, the video resizing does not shrink below the specified values.

Copyright © 2001 -2024 Syncfusion Inc.



Rename video before inserting

You can use the insertVideoSettings property, to specify the server handler to upload the selected video. Then by binding the fileUploadSuccess event, you can receive the modified file name from the server and update it in the Rich Text Editor's insert video dialog.

`

<template>

<ejs-richtexteditor ref="rteObj" :height = "340" :insertVideoSettings = "insertVideoSettings" :toolbarSettings = "toolbarSettings" :fileUploadSuccess = "onFileUploadSuccess">

The Rich Text Editor is WYSIWYG ("what you see is what you get") editor useful to create and edit content, and return the valid HTML markup or markdown of the content

</ejs-richtexteditor>

</template>

<script>

import Vue from "vue";

```
import { RichTextEditorPlugin, Toolbar, Link, Video, HtmlEditor, QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
data: function() {
return {
toolbarSettings: {
items: ['Video']
},
insertVideoSettings: {
saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/Rename",
path: "[SERVICEHOSTEDPATH]/Files/"
}
};
},
methods: {
onFileUploadSuccess (args) {
alert("Get the new file name here");
if (args.e.currentTarget.getResponseHeader('name') != null) {
args.file.name = args.e.currentTarget.getResponseHeader('name');
let filename = document.querySelectorAll(".e-file-name")[0];
filename.innerHTML = args.file.name.replace(document.querySelectorAll(".e-file-type")[0].innerHTML, '');
filename.title = args.file.name;
}
}
},
provide:{
richtexteditor:[Toolbar, Link, Video, HtmlEditor, QuickToolbar]
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
```

```
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import '../node_modules/@syncfusion/ej2-richtexteditor/styles/material.css';
@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-gantt/styles/material.css";
</style>
`
```

To configure server-side handler, refer to the below code.

```c#
int x = 0;
string file;
[AcceptVerbs("Post")]
public void Rename()
{
try
{
var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
fileName = httpPostedFile.FileName;
if (httpPostedFile != null)
{
var fileSave = System.Web.HttpContext.Current.Server.MapPath("~/Files");
if (!Directory.Exists(fileSave))
{
Directory.CreateDirectory(fileSave);
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
var fileName = Path.GetFileName(httpPostedFile.FileName);
var fileSavePath = Path.Combine(fileSave, fileName);
while (System.IO.File.Exists(fileSavePath))
{
fileName = "rteFiles" + x + "-" + fileName;
fileSavePath = Path.Combine(fileSave, fileName);
x++;
}
if (!System.IO.File.Exists(fileSavePath))
{
httpPostedFile.SaveAs(fileSavePath);
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.Headers.Add("name", fileName);
Response.ContentType = "application/json; charset=utf-8";
Response.StatusDescription = "File uploaded succesfully";
Response.End();
}
}
}
catch (Exception e)
{
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusCode = 204;
Response.Status = "204 No Content";
Response.StatusDescription = e.Message;
Response.End();
}
}
`
```

Copyright © 2001 -2024 Syncfusion Inc.

Upload video with authentication

You can add additional data with the video uploaded from the Rich Text Editor on the client side, which can even be received on the server side by using the fileUploading event and its customFormData argument, you can pass parameters to the controller action. On the server side, you can fetch the custom headers by accessing the form collection from the current request, which retrieves the values sent using the POST method.

By default, it doesn't support the UseDefaultCredentials property, you can manually append the default credentials with the upload request.

```
<template>
<div>
<div class="control-section">
<div class="sample-container">
<div class="default-section">
<ejs-richtexteditor ref="rteObj" :height = "340" :insertVideoSettings = "insertVideoSettings" :toolbarSettings = "toolbarSettings" :fileUploading = "onFileUpload">
</ejs-richtexteditor>
</div>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Video, HtmlEditor, QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
data: function() {
return {
toolbarSettings: {
items: ['Video']
},
insertVideoSettings: {
saveUrl: "[SERVICE*HOSTED*PATH]/api/uploadbox/SaveFiles",
```

Copyright © 2001 -2024 Syncfusion Inc.

```
path: "[SERVICE*HOSTED*PATH]/Files/"
}
};
},
methods: {
onFileUpload (args) {
alert("Get the new file name here");
var accessToken = "Authorization_token";
// adding custom Form Data
args.customFormData = [{ 'Authorization': accessToken }];
}
},
provide:{
richtexteditor:[Toolbar, Link, Video, HtmlEditor, QuickToolbar]
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import '../node_modules/@syncfusion/ej2-richtexteditor/styles/material.css';
@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-gantt/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

`

`c#

public void SaveFiles(IList<IFormFile> UploadFiles)

{

string currentPath = Request.Form["Authorization"].ToString();

}

`

See Also

- How to edit the quick toolbar settings
- How to use the link editing option in the toolbar items

Link in Vue Rich text editor component

A hyperlink can be insert into the editor for quick access to the related information. The hyperlink itself can be a text or an image.

Insert link

Point the cursor anywhere within the editor where you would like to insert the link. It is also possible to select a text or an image within the editor and can be converted to the hyperlink. Click the Insert HyperLink tool on the toolbar. The Insert Link Dialog will be open. The dialog has the following options.

`ts

To use use image and link tool, inject `ImageService, LinkService` in the provider section of `AppModule`.

`

| Options | Description |

|---|---|

| Web Address | Type or paste the destination for the link you are creating |

| Display Text | Type or edit the required text that you want to display text for the link|

| Tooltip | To display additional helpful information when you place the pointer on the hyperlink, type the required text in the “Tooltip” field. |

| Open Link in New Window | Specify whether, the given link will be open in new window or not |

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj"
:toolbarSettings="toolbarSettings"><p>Rich
        Text Editorallows to insert images from online source as well as
local
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            computer where you want to insert the image in your content.</p>
            <p><b>Get started Quick Toolbar to click on the image</b></p>
            <p>It is possible to add custom style on the selected image
inside the Rich Text Editor through quick toolbar.</p>
        </ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Image,  Link, HtmlEditor } from
"@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
    data: function() {
        return {
            toolbarSettings: {
                items: ['CreateLink']
            }
        };
    },
    provide:{
        richtexteditor:[Toolbar, Image,  Link, HtmlEditor]
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs5" %}

The Rich Text Editor link tool validates the URLs, as you type in the Web Address. URLs considered invalid will be highlighted with red color by clicking the insert button in the Insert Link dialog.

Remove link

If you want to remove a hyperlink from a text or image, select the text or image with the hyperlink and click “Remove Hyperlink” tool from toolbar. It will keep the text or image.

Auto-link

When you type URL and Enter key to the Rich Text Editor, the typed URL will be automatically changed into the hyperlink.

Copyright © 2001 -2024 Syncfusion Inc.

Manipulation

Add the custom tools on the selected link inside the Rich Text Editor through the quick toolbar.

The quick toolbar for the Link has the following options.

| Tools | Description |

|---|---|

| Open | The given link page, will be open in new window. |

| Edit Link | Used to edit the link in the Rich Text Editor content. |

| Remove Link | Used to remove link from the content of Rich Text Editor. |

| Custom Tool | Used to add the custom options in the quick toolbar. |

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj"
:toolbarSettings="toolbarSettings"><p>Rich
        Text Editor allows to insert images from online source as well as
local
            computer where you want to insert the image in your content.</p>
            <p><b>Get started Quick Toolbar to click on the image</b></p>
            <p>It is possible to add custom style on the selected image
inside the Rich Text Editor through quick toolbar.</p>
        </ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Image,  Link, HtmlEditor,
QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
    data: function() {
        return {
            toolbarSettings: {
                items: ['CreateLink']
            }
        };
    },
    provide:{
        richtexteditor:[Toolbar, Image,  Link, HtmlEditor, QuickToolbar]
    }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs6" %}

See Also

- How to edit the quick toolbar settings
- How to insert image link editing option in the toolbar items

Table in Vue Rich text editor component

Rich Text Editor allows to insert table of content in edit panel and provide options to add, edit, and remove the table as well as perform other table related action. For inserting the table to the Rich Text Editor, the following list of options have been provided in the tableSettings

| Options | Description | Default Value |

|---|---|---|

| minWidth | Sets the default minWidth of the table. | 0 |

| maxWidth | Sets the default maxWidth of the table. | null |

| resize | Enable resize feature in table.| true |

| styles | This is an array of key value pair, on each pair, key should be name of styling and value is class name. this list will be shown on quick toolbar options to change the styles of table on designing like dashed, double bordered. | TableStyleItems |

| width | Sets the default width of the table. | 100% |

Insert table

Using the table toolbar option, select a number of rows and columns to be inserted over the table grid and insert table into Rich Text Editor content using the mouse.

Tables can also be inserted through the Insert Table option in the pop-up where the number of rows and columns can be provided manually, and this is the default way in devices.

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj" :height="340"
:toolbarSettings="toolbarSettings"><p>The Rich Text Editor component is
WYSIWYG ("what you see is what you get") editor that provides the best user
experience to create and update the content.
                Users can format their content using standard toolbar
commands.</p>
                <p><b>Key features:</b></p>
                <ul><li><p>Provides < IFRAME > and < DIV > modes</p></li>
                <li><p>Capable of handling markdown editing.</p></li>
                <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
                <li><p>Provides a fully customizable toolbar.</p></li>
                <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
                <li><p>Supports third-party library integration.</p></li>
                <li><p>Allows preview of modified content before saving
it.</p></li>
                <li><p>Handles images, hyperlinks, video, hyperlinks,
uploads, etc.</p></li>
                </ul></ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar, Table } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
            toolbarSettings: {
                items: ['CreateTable']
            }
        };
    },
    provide:{
        richtexteditor:[Toolbar, Link, Image, HtmlEditor, QuickToolbar,
Table]
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

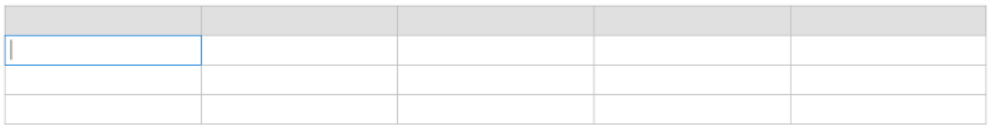
{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs15" %}

Quick Toolbar

Quick toolbar is opened by clicking the table. It has different sets of commands to be performed on the table which increases the feasibility to edit the table easily.

Table Header

Table Header command is available with quick toolbar option through which the header row can be added or removed from the inserted table. The following image illustrates the table header.



Insert Rows

Rows can be inserted above or below the required table cell through the quick toolbar. Also, focused row can be deleted. The following screenshot shows the available options of the row item.

Insert Columns

Columns can be inserted to the left or right side of the required table cell through the quick toolbar. Also, the focused column can be deleted. The following screenshot shows the available options of the column item.

Set Color

The background color can be set for each table cell through the background color command available with quick toolbar.

Delete Table

Using the delete item in the quick toolbar, users can delete the entire table.

Vertical Align

Text inside the table can be aligned to top, middle, or bottom using the `tableCellVerticalAlign` tool of the quick toolbar.

Horizontal Align

Text inside the table can be aligned left, right, or center using the `tableCellHorizontalAlign` tool of the quick toolbar.

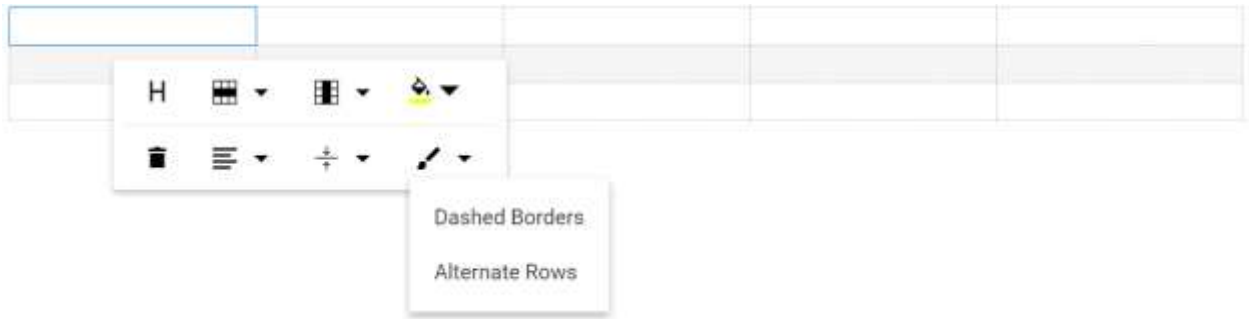
Table Styles

Table styles provided for class name should be appended to a table element. It helps to design the table in specific CSS styles when inserting in the editor.

By Default, provides Dashed border and Alternate rows.

Dashed border: Applies the dashed border to the table.

Alternate border: Applies the alternative background to the table.



Copyright © 2001 -2024 Syncfusion Inc.

Table Properties

Sets the default width of the table when it is inserted in the Rich Text Editor using the width of `tableSettings`.

Using the quick toolbar, users can change the width, cell padding, and cell spacing in the selected table using the properties option.

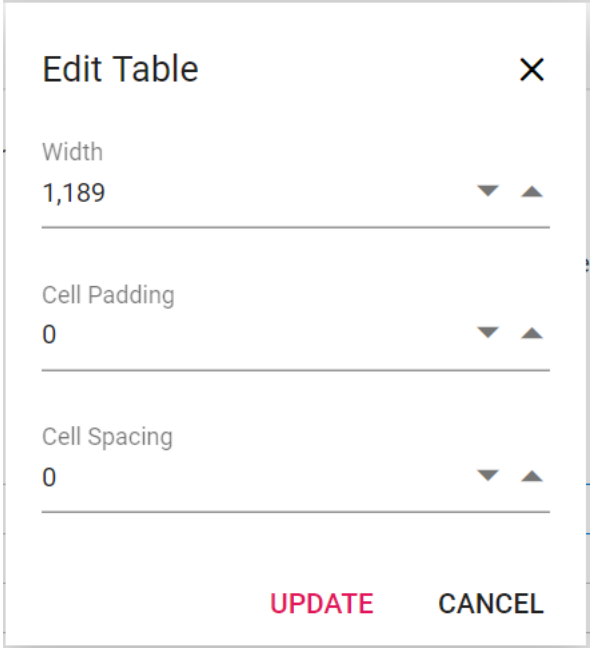


Table cell merge and split

The Rich Text Editor allows users to change the appearance of the tables by splitting or merging the table cells.

`TableCell` item should be configured in the Table quickToolbarSettings Property to show the merge/split icons while selecting the table cells

Table cell merge

The table cell merge feature allows you to merge two or more row and column cells into a single cell with its contents.

The following image explains the table merge action.

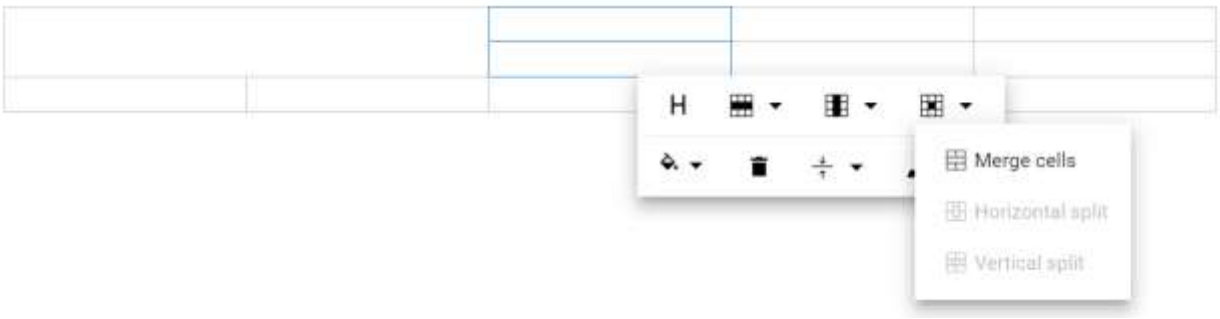
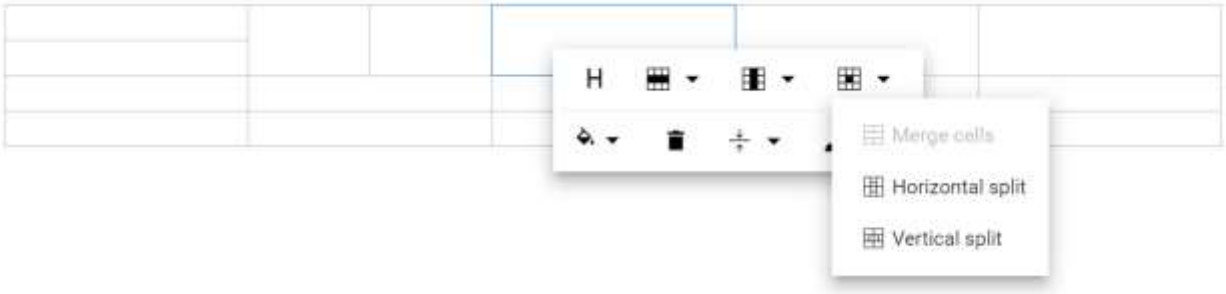


Table cell split

The table cell split feature allows you to a selected cell can be split both horizontally and vertically.

The following image explains the table split action.



Emoji Picker in Vue RichTextEditor component

An emoji picker is a tool that allows users to add emojis or emoticons to their text easily. Typically, it is a small window or panel that displays a variety of emojis arranged in different categories, such as smileys, animals, food, and so on. Users can select the desired emoji by clicking on it or by typing its name in a search bar.

Enabling the toolbar option and custom emojis.

Add the `EmojiPicker` tool to the toolbar of the RichTextEditor by utilizing the `toolbarSettings` items property.

By default, a predefined set of emojis is configured. However, you can customize these icons according to your needs. To achieve this, utilize the emojiPickerSettings property.

```
<template>
<div>
<div class="control-section">
<div class="sample-container">
<div class="default-section">
<ejs-richtexteditor ref="rteObj" :height="340" :emojiPickerSettings="emojiPickerSettings"
:toolbarSettings="toolbarSettings">
</ejs-richtexteditor>
</div>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor, QuickToolbar, EmojiPicker } from
"@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
```

```
export default {
data: function() {
return {
toolbarSettings: {
items: ['Bold', 'Italic', 'Underline', '|', 'Formats', 'Alignments', 'OrderedList',
'UnorderedList', '|', 'CreateLink', 'Image', '|', 'SourceCode', 'EmojiPicker', '|', 'Undo', 'Redo']
},
emojiPickerSettings: {
iconsSet: [{name: 'Smilies & People', code: '1F600', iconCss: 'e-emoji',
icons: [{ code: '1F600', desc: 'Grinning face' },
{ code: '1F603', desc: 'Grinning face with big eyes' },
{ code: '1F604', desc: 'Grinning face with smiling eyes' },
{ code: '1F606', desc: 'Grinning squinting face' },
{ code: '1F605', desc: 'Grinning face with sweat' },
{ code: '1F602', desc: 'Face with tears of joy' },
{ code: '1F923', desc: 'Rolling on the floor laughing' },
{ code: '1F60A', desc: 'Smiling face with smiling eyes' }]
}, {
name: 'Animals & Nature', code: '1F435', iconCss: 'e-animals',
icons: [{ code: '1F436', desc: 'Dog face' },
{ code: '1F431', desc: 'Cat face' },
{ code: '1F42D', desc: 'Mouse face' },
{ code: '1F439', desc: 'Hamster face' },
{ code: '1F430', desc: 'Rabbit face' },
{ code: '1F98A', desc: 'Fox face' }]
}, {
name: 'Food & Drink', code: '1F347', iconCss: 'e-food-and-drinks',
icons: [{ code: '1F34E', desc: 'Red apple' },
{ code: '1F34C', desc: 'Banana' },
{ code: '1F347', desc: 'Grapes' },
{ code: '1F353', desc: 'Strawberry' },
{ code: '1F35E', desc: 'Bread' },
{ code: '1F950', desc: 'Croissant' },
```

```
{ code: '1F955', desc: 'Carrot' },
{ code: '1F354', desc: 'Hamburger' }]
}, {
name: 'Activities', code: '1F383', iconCss: 'e-activities',
icons: [{ code: '26BD', desc: 'Soccer ball' },
{ code: '1F3C0', desc: 'Basketball' },
{ code: '1F3C8', desc: 'American football' },
{ code: '26BE', desc: 'Baseball' },
{ code: '1F3BE', desc: 'Tennis' },
{ code: '1F3D0', desc: 'Volleyball' },
{ code: '1F3C9', desc: 'Rugby football' }]
}, {
name: 'Travel & Places', code: '1F30D', iconCss: 'e-travel-and-places',
icons: [{ code: '2708', desc: 'Airplane' },
{ code: '1F697', desc: 'Automobile' },
{ code: '1F695', desc: 'Taxi' },
{ code: '1F6B2', desc: 'Bicycle' },
{ code: '1F68C', desc: 'Bus' }]
}, {
name: 'Objects', code: '1F507', iconCss: 'e-objects', icons: [{ code: '1F4A1', desc: 'Light bulb' },
{ code: '1F526', desc: 'Flashlight' },
{ code: '1F4BB', desc: 'Laptop computer' },
{ code: '1F5A5', desc: 'Desktop computer' },
{ code: '1F5A8', desc: 'Printer' },
{ code: '1F4F7', desc: 'Camera' },
{ code: '1F4F8', desc: 'Camera with flash' },
{ code: '1F4FD', desc: 'Film projector' }]
}, {
name: 'Symbols', code: '1F3E7', iconCss: 'e-symbols', icons: [{ code: '274C', desc: 'Cross mark' },
{ code: '2714', desc: 'Check mark' },
{ code: '26A0', desc: 'Warning sign' },
{ code: '1F6AB', desc: 'Prohibited' },
{ code: '2139', desc: 'Information' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ code: '267B', desc: 'Recycling symbol' },
{ code: '1F6AD', desc: 'No smoking' }]
}]
}
};
},
provide:{
richtexteditor:[Toolbar, Link, Image, HtmlEditor, QuickToolbar, EmojiPicker]
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-richtexteditor/styles/material.css';
</style>
`
```

Additionally, you have the option to customize the icons of toolbar items using the iconCss and code properties. The iconCSS property allows you to define a custom CSS class for the toolbar item icon, while the code property enables you to specify the Unicode character code for the icon.

When both iconCSS and code properties are provided, the iconCSS property takes precedence in determining the appearance of the toolbar item icon.

Additionally, you have the option to enhance the user experience by implementing a filtering feature for efficiently managing a large dataset of emojis. By setting the showSearchBox property to true (which is the default value), users will be able to utilize a search box to filter the displayed emojis according to their preferences.

The following code example shows how to add the emoji picker tool in the RichTextEditor.

APP.VUE

```
<template>
```

```
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj" :toolbarSettings="toolbarSettings"
>
            <p>An emoji picker in a Rich Text Editor is a tool that allows
users to easily add emojis or emoticons to their text.</p>
            <p>Typically, it is a small window or panel that displays a
variety of emojis, arranged in different categories, such as smileys,
animals, food, and so on. Users can select the desired emoji by clicking on
it or by typing its name in a search bar.</p>
        </ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar, EmojiPicker } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
            toolbarSettings: {
                items: ['Bold', 'Italic', 'Underline', '|', 'Formats',
'Alignments', 'OrderedList',
                'UnorderedList', '|', 'CreateLink', 'Image', '|',
'SourceCode', 'EmojiPicker', '|', 'Undo', 'Redo']
            },
        };
    },
    provide:{
        richtexteditor:[Toolbar, Link, Image, HtmlEditor, QuickToolbar,
EmojiPicker]
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-
richtexteditor/styles/material.css';
</style>
```

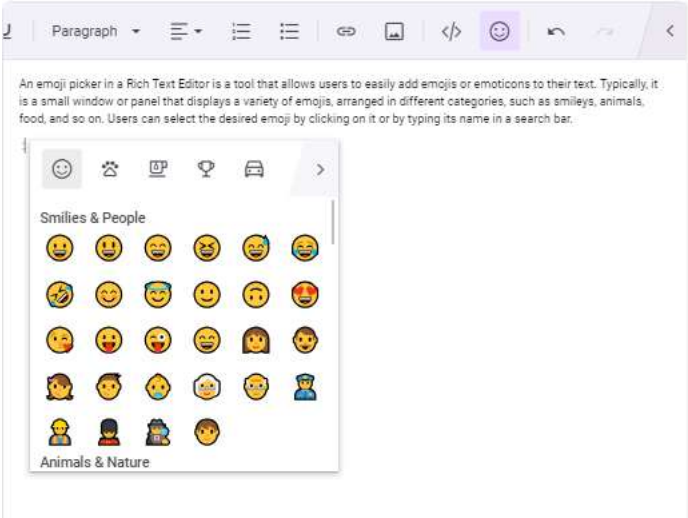
{% previewsample "page.domainurl/code-snippet/rich-text-editor/emoji-picker-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Rich Text Editor features are segregated into individual feature-wise modules. To use emojis, inject the EmojiPicker module in provide section.

Using the shortcut key to open the emoji picker

Quickly access the emoji picker by pressing the colon (:) key while typing a word prefix in an editor, allowing instant emoji selection and display. Moreover, continue typing in the editor after the colon (:) to filter and refine your search for the desired emojis.



Navigating and selecting emojis using the keyboard

The emoji picker popup offers keyboard navigation options, allowing you to move the emoji focus from one emoji to another. The following keys are used for navigation:

Arrow keys: Use the arrow keys (up, down, left, right) to move the emoji focus in the corresponding direction.

Enter: Press Enter key to select the currently focused emoji.

Escape: Press Escape to close the emoji picker popup without selecting an emoji.

Markdown in Vue Rich text editor component

When you format the word in Markdown format, you should add Markdown syntax to the word to indicate the words and phrases that looks different from each other.

Rich Text Editor supports markdown editing when the editorMode set as **markdown** and using both *keyboard interaction* and *toolbar action*, you can apply the formatting to text

To use quick Markdown editing feature, inject MarkdownEditorService in the provider section of AppModule.

Supported Commands

The Vue Markdown editor supports the following commands to format the markdown content:

|Commands|Syntax| Description |

|--------|--|------------|

| Bold | Sample content for **bold text**. | For bold, add or to front and back of the text. | For order list, precede each line with a number. |

Copyright © 2001 -2024 Syncfusion Inc.

| Italic | Sample content for *Italic text*. | For Italic, add * or _ to front and back of the text. |

| Bold and Italics | Sample content for ***bold and Italic text***. | For bold and Italics, add * to the front and back of the text. |

| Heading 1 | # Heading 1 content | For heading 1, add # to start of the line. |

| Heading 2 | ## Heading 2 content | For heading 2, add ## to start of the line. |

| Heading 3 | ### Heading 3 content | For heading 3, add ### to start of the line. |

| Heading 4 | #### Heading 4 content | For heading 4, add #### to start of the line. |

| Heading 5 | ##### Heading 5 content | For heading 5, add ##### to start of the line. |

| Heading 6 | ###### Heading 6 content | For heading 6, add ###### to start of the line. |

| Line Break | First line
Second line | For line break, press enter two times (or) add <br> in between the first and the second line. |

| Blockquotes | > Blockquotes text | For blockquotes, add > to start of the line. |

| Strike Through | Sample content for ~~strike through text~~. | For strike through, add ~~ to front and back of the text. |

| Code (Single line) | \Single line code\ | For single line code, add ` to front and back of the text. |

| Code block (Multi Line) | \\\
Multi line code text
Multi line code text
\\\ | For multiple line code, add \\` in the new line before and after the content. |

| Subscript | _{Subscript text} | For subscript, add _{to the front and} to the back of the text. |

| Superscript | ^{Superscript text} | For superscript, add ^{to the front and} to the back of the text. |

| Ordered List | 1. First
1. Second | For ordered list, preceding one or more lines of text with 1. |

| Unordered List | *First
* second | For unordered list, preceding one or more lines of text with *. |

| Links | **Link text without title text**
`` Link text ``
 Link text with title text
[Link text](URL , “title text”) | Create an inline link by wrapping link text in brackets [ ], and then wrapping the URL as first parameter and title as second parameter in the parentheses ().
Note: The title text is optional, if needed it can be given manually.|

Table		Heading 1	Heading 2
Col A1	Col A2
Col B1	Col B2		Create a table using the pipes and underscores as given in the syntax to create 2 x 2 table.

| Horizontal Line | ***(three asterix in new line)
(or)
 (three underscores in new line) | For horizontal line, add or* to the start of the new line. |

| Image | | Create an image by wrapping the image source in parentheses (). |

| Image with alternate text | ![alternate text](URL path) | Create an image with alternate text by wrapping an alternative text in brackets [], and then link of the image source in parentheses

Copyright © 2001 -2024 Syncfusion Inc.

().
Note: When inserting the image using toolbar, the alternate text cannot be provided that needs to be given manually. |

| Escape tick marks supported | Sample text content with **bold and** not bold **text can be in the same line.** | In the syntax, the whole content is made as bold where the content not bold can be made as normal text by adding the bold syntax to the start and end of the respective text. Likewise you can do the same for various inline commands. |

| Escape Character | \(any syntax) | Escape any markdown syntax by prefix \ to the syntax.
Example:
Bold text|

| HTML Entities | Copyright: © - ©
Trade mark: ™ - ™
Registered: ® - ®
Ampersand: & - &
Less than: < - <
Greater than: > - > | For HTML entities, add & and ; to the front and back of the respective entities. |

The above listed commands alone are supported in Syncfusion Markdown editor. For other unsupported commands, you can achieve using the HTML tags in Markdown editor. The foot notes, definitions, math, and check list markdown syntax are also not supported.

Markdown to HTML

The Rich Text Editor allows you to preview markdown changes immediately using preview. In this sample, the third-party library Marked is used to convert markdown into HTML content.

This sample demonstrates how to preview markdown changes in Rich Text Editor. Type or edit the display text, and apply format to view the preview of markdown. The actionComplete event can be used to convert Markdown to HTML.

APP.VUE

```
<template>
  <div id="defaultRTE">
    <ejs-richtexteditor id="preview" ref="rteInstance"
:toolbarSettings="toolbarSettings" :created="created"
:actionComplete='actionComplete' :editorMode="editorMode" :height="height">
In Rich Text Editor , you click the toolbar buttons to format the words and
the changes
are visible immediately.
Markdown is not like that. When you format the word in Markdown format, you
need to
add Markdown syntax to the word to indicate which words
and phrases should look different from each other
Rich Text Editor supports markdown editing when the editorMode set as
**markdown** and using both *keyboard interaction* and *toolbar action*, you
can apply the formatting to text.
We can add our own custom formation syntax for the Markdown formation,
[sample link](https://ej2.syncfusion.com/home/).
The third-party library <b>Marked</b> is used in this sample to convert
markdown into HTML content.
      </ejs-richtexteditor>
  </div>
</template>
    <style>
    .e-richtexteditor .e-rte-content .e-content{
```

```
        min-height: 150px;
    }
    .e-richtexteditor .e-rte-content textarea.e-content {
        float: left;
        border-right: 1px solid rgba(0, 0, 0, 0.12);
    }
    .e-richtexteditor .e-rte-content {
        overflow: hidden;
    }
    .e-md-preview::before {
        content: '\e345';
    }
    .e-rte-content .e-content.e-pre-source {
        width: 100%;
    }
    .e-icon-btn.e-active .e-md-preview.e-icons::before {
        content: '\e350';
    }
</style>
<script>
import Vue from "vue";
import { Browser, addClass, removeClass, isNullOrUndefined } from
"@syncfusion/ej2-base";
import { RichTextEditorPlugin, Toolbar, Link, Image, MarkdownEditor } from
"@syncfusion/ej2-vue-richtexteditor";
import { createElement, KeyboardEventArgs } from '@syncfusion/ej2-vue-base';
Vue.use(RichTextEditorPlugin);
export default {
    data: function() {
        return {
            textArea: '',
            height: '300px',
        toolbarSettings: {
            items: ['Bold', 'Italic', 'StrikeThrough', '|', 'Formats',
'OrderedList', 'UnorderedList', '|', 'CreateLink', 'Image', '|',
                { tooltipText: 'Preview', template: '<button id="preview-
code" class="e-tbar-btn e-control e-btn e-icon-btn">' +
                    '<span class="e-btn-icon e-md-preview e-
icons"></span></button>' },
                { tooltipText: 'Split Editor', template: '<button
id="MD_Preview" class="e-tbar-btn e-control e-btn e-icon-btn">' +
                    '<span class="e-btn-icon e-view-side e-
icons"></span></button>' }, 'FullScreen', '|', 'Undo', 'Redo']
        },
        editorMode: 'Markdown',
        };
    },
    methods: {
        created: function() {
            var script = document.createElement('script');
            script.src =
'https://cdn.jsdelivr.net/npm/marked/marked.min.js';
            document.head.appendChild(script);
            this.textArea =
this.$refs.rteInstance.$el.parentNode.querySelector('.e-content');
            this.textArea.onkeyup = (Event) => {
                this.markDownConversion();
```

```
            };
            document.getElementById('preview-code').onclick = () => {
                this.fullPreview({ mode: true, type: 'preview'});
                 if (event.target.parentElement.classList.contains('e-
active')) {

this.$refs.rteInstance.ej2Instances.disableToolbarItem(['Bold', 'Italic',
'StrikeThrough', 'Formats', 'OrderedList',
                    'UnorderedList', 'CreateLink', 'Image']);

event.target.parentElement.parentElement.nextElementSibling.classList.add('e
-overlay');
                } else {

this.$refs.rteInstance.ej2Instances.enableToolbarItem(['Bold', 'Italic',
'StrikeThrough', 'Formats', 'OrderedList',
                    'UnorderedList', 'CreateLink', 'Image']);

event.target.parentElement.parentElement.nextElementSibling.classList.remove
('e-overlay');
                }
            };
            document.getElementById('MD_Preview').onclick = () => {
                if (this.$refs.rteInstance.$el.classList.contains('e-rte-
full-screen')) {
                    this.fullPreview({ mode: true, type: '' });
                }
                document.getElementById('preview-code').classList.remove('e-
active');
                this.$refs.rteInstance.ej2Instances.showFullScreen();
            };
        },
        markDownConversion: function(){
        if (document.getElementById('MD_Preview').classList.contains('e-
active')) {
            var id = this.$refs.rteInstance.ej2Instances.getID() + 'html-
view';
            var htmlPreview =
this.$refs.rteInstance.$el.parentNode.querySelector('#' + id);
            htmlPreview.innerHTML = marked(this.textArea.value);
        }
    },
    actionComplete: function(e) {
        var mdsource = document.getElementById('preview-code');
        var mdSplit = document.getElementById('MD_Preview');
        var id = this.$refs.rteInstance.ej2Instances.getID() + 'html-view';
        var htmlPreview =
this.$refs.rteInstance.$el.parentNode.querySelector('#' + id);
            if (e.targetItem === 'Maximize' && isNullOrUndefined(e.args)) {
                this.fullPreview({ mode: true, type: '' });
            } else if (!mdSplit.parentElement.classList.contains('e-
overlay')) {
                if (e.targetItem === 'Minimize') {
                this.textArea.style.display = 'block';
                this.textArea.style.width = '100%';
                if (htmlPreview) { htmlPreview.style.display = 'none'; }
                mdSplit.classList.remove('e-active');
```

```
                mdsource.classList.remove('e-active');
                }
            this.markDownConversion();
            }
        },
        fullPreview: function(event){
            var mdsource = document.getElementById('preview-code');
            var mdSplit = document.getElementById('MD_Preview');
            var id = this.$refs.rteInstance.ej2Instances.getID() + 'html-
view';
            var htmlPreview =
this.$refs.rteInstance.$el.parentNode.querySelector('#' + id);
        if ((mdsource.classList.contains('e-active') ||
mdSplit.classList.contains('e-active')) && event.mode) {
            mdsource.classList.remove('e-active');
            mdSplit.classList.remove('e-active');
            this.textArea.style.display = 'block';
            this.textArea.style.width = '100%';
            htmlPreview.style.display = 'none';
        } else {
            mdsource.classList.add('e-active');
             mdSplit.classList.add('e-active');
            if (!htmlPreview) {
                htmlPreview = document.createElement('div');
                htmlPreview.id = id;
                htmlPreview.setAttribute('class', 'e-content');
                this.textArea.parentNode.appendChild(htmlPreview);
            }
            if (event.type === 'preview') {
                this.textArea.style.display = 'none';
                htmlPreview.classList.add('e-pre-source');
            } else {
                htmlPreview.classList.remove('e-pre-source');
                this.textArea.style.width = '50%';
            }
            htmlPreview.style.display = 'block';
            htmlPreview.innerHTML =
marked(this.$refs.rteInstance.ej2Instances.contentModule.getEditPanel().valu
e);
            mdsource.parentElement.title = 'Code View';
        }
    }
    },
    provide:{
        richtexteditor:[Toolbar, Link, Image, MarkdownEditor]
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs7" %}

Table

Rich Text Editor allows to insert Markdown table in edit panel with 2 X 2 rows and columns along with the heading. To use table tool, add the `CreateTable` item in toolbar items.

Insert table

To insert the table in Rich Text Editor, click the `table` toolbar option to insert the table into Rich Text Editor content and this is the default way in all the devices.

Please refer the below sample and code snippets to add the table in Markdown editor

APP.VUE

```
<template>
  <div id="defaultRTE">
    <ejs-richtexteditor id="preview" ref="rteInstance"
:toolbarSettings="toolbarSettings" :created="created"
:actionComplete='actionComplete' :editorMode="editorMode" :height="height">
In Rich Text Editor , you click the toolbar buttons to format the words and
the changes
are visible immediately.
Markdown is not like that. When you format the word in Markdown format, you
need to
add Markdown syntax to the word to indicate which words
and phrases should look different from each other
Rich Text Editor supports markdown editing when the editorMode set as
**markdown** and using both *keyboard interaction* and *toolbar action*, you
can apply the formatting to text.
We can add our own custom formation syntax for the Markdown formation,
[sample link](https://ej2.syncfusion.com/home/).
The third-party library <b>Marked</b> is used in this sample to convert
markdown into HTML content.
      </ejs-richtexteditor>
  </div>
</template>
    <style>
    .e-richtexteditor .e-rte-content .e-content{
        min-height: 150px;
    }
    .e-richtexteditor .e-rte-content textarea.e-content {
        float: left;
        border-right: 1px solid rgba(0, 0, 0, 0.12);
    }
    .e-richtexteditor .e-rte-content {
        overflow: hidden;
    }
    .e-md-preview::before {
        content: '\e345';
    }
```

```
    .e-rte-content .e-content.e-pre-source {
        width: 100%;
    }
    .e-icon-btn.e-active .e-md-preview.e-icons::before {
        content: '\e350';
    }
</style>
<script>
import Vue from "vue";
import { Browser, addClass, removeClass, isNullOrUndefined } from
"@syncfusion/ej2-base";
import { RichTextEditorPlugin, Toolbar, Link, Image, MarkdownEditor } from
"@syncfusion/ej2-vue-richtexteditor";
import { createElement, KeyboardEventArgs } from '@syncfusion/ej2-vue-base';
Vue.use(RichTextEditorPlugin);
export default {
    data: function() {
        return {
            textArea: '',
            height: '300px',
        toolbarSettings: {
            items: ['Bold', 'Italic', 'StrikeThrough', '|', 'Formats',
'OrderedList', 'UnorderedList', '|', 'CreateLink', 'Image', 'CreateTable',
'|',
                { tooltipText: 'Preview', template: '<button id="preview-
code" class="e-tbar-btn e-control e-btn e-icon-btn">' +
                    '<span class="e-btn-icon e-md-preview e-
icons"></span></button>' },
                { tooltipText: 'Split Editor', template: '<button
id="MD_Preview" class="e-tbar-btn e-control e-btn e-icon-btn">' +
                    '<span class="e-btn-icon e-view-side e-
icons"></span></button>' }, 'FullScreen', '|', 'Undo', 'Redo']
        },
        editorMode: 'Markdown',
        };
    },
    methods: {
        created: function() {
            var script = document.createElement('script');
            script.src =
'https://cdn.jsdelivr.net/npm/marked/marked.min.js';
            document.head.appendChild(script);
            this.textArea =
this.$refs.rteInstance.$el.parentNode.querySelector('.e-content');
            this.textArea.onkeyup = (Event) => {
                this.markDownConversion();
            };
            document.getElementById('preview-code').onclick = () => {
                this.fullPreview({ mode: true, type: 'preview'});
                 if (event.target.parentElement.classList.contains('e-
active')) {

this.$refs.rteInstance.ej2Instances.disableToolbarItem(['Bold', 'Italic',
'StrikeThrough', 'Formats', 'OrderedList',
                    'UnorderedList', 'CreateLink', 'Image',
'CreateTable']);
```

```
event.target.parentElement.parentElement.nextElementSibling.classList.add('e
-overlay');
                } else {

this.$refs.rteInstance.ej2Instances.enableToolbarItem(['Bold', 'Italic',
'StrikeThrough', 'Formats', 'OrderedList',
                    'UnorderedList', 'CreateLink', 'Image',
'CreateTable']);

event.target.parentElement.parentElement.nextElementSibling.classList.remove
('e-overlay');
                }
            };
            document.getElementById('MD_Preview').onclick = () => {
                if (this.$refs.rteInstance.$el.classList.contains('e-rte-
full-screen')) {
                    this.fullPreview({ mode: true, type: '' });
                }
                document.getElementById('preview-code').classList.remove('e-
active');
                this.$refs.rteInstance.ej2Instances.showFullScreen();
            };
        },
        markDownConversion: function(){
        if (document.getElementById('MD_Preview').classList.contains('e-
active')) {
            var id = this.$refs.rteInstance.ej2Instances.getID() + 'html-
view';
            var htmlPreview =
this.$refs.rteInstance.$el.parentNode.querySelector('#' + id);
            htmlPreview.innerHTML = marked(this.textArea.value);
        }
    },
    actionComplete: function(e) {
        var mdsource = document.getElementById('preview-code');
        var mdSplit = document.getElementById('MD_Preview');
        var id = this.$refs.rteInstance.ej2Instances.getID() + 'html-view';
        var htmlPreview =
this.$refs.rteInstance.$el.parentNode.querySelector('#' + id);
            if (e.targetItem === 'Maximize' && isNullOrUndefined(e.args)) {
                this.fullPreview({ mode: true, type: '' });
            } else if (!mdSplit.parentElement.classList.contains('e-
overlay')) {
                if (e.targetItem === 'Minimize') {
                this.textArea.style.display = 'block';
                this.textArea.style.width = '100%';
                if (htmlPreview) { htmlPreview.style.display = 'none'; }
                mdSplit.classList.remove('e-active');
                mdsource.classList.remove('e-active');
                }
            this.markDownConversion();
            }
        },
        fullPreview: function(event){
            var mdsource = document.getElementById('preview-code');
             var mdSplit = document.getElementById('MD_Preview');
```

```
            var id = this.$refs.rteInstance.ej2Instances.getID() + 'html-
view';
            var htmlPreview =
this.$refs.rteInstance.$el.parentNode.querySelector('#' + id);
        if ((mdsource.classList.contains('e-active') ||
mdSplit.classList.contains('e-active')) && event.mode) {
            mdsource.classList.remove('e-active');
            mdSplit.classList.remove('e-active');
            this.textArea.style.display = 'block';
            this.textArea.style.width = '100%';
            htmlPreview.style.display = 'none';
        } else {
            mdsource.classList.add('e-active');
             mdSplit.classList.add('e-active');
            if (!htmlPreview) {
                htmlPreview = document.createElement('div');
                htmlPreview.id = id;
                htmlPreview.setAttribute('class', 'e-content');
                this.textArea.parentNode.appendChild(htmlPreview);
            }
            if (event.type === 'preview') {
                this.textArea.style.display = 'none';
                htmlPreview.classList.add('e-pre-source');
            } else {
                htmlPreview.classList.remove('e-pre-source');
                this.textArea.style.width = '50%';
            }
            htmlPreview.style.display = 'block';
            htmlPreview.innerHTML =
marked(this.$refs.rteInstance.ej2Instances.contentModule.getEditPanel().valu
e);
            mdsource.parentElement.title = 'Code View';
        }
    }
    },
    provide:{
        richtexteditor:[Toolbar, Link, Image, MarkdownEditor]
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs8" %}

Copyright © 2001 -2024 Syncfusion Inc.

Changing table constants

The Markdown table constants can be changed for the table heading and the column names.

APP.VUE

```
<template>
  <div id="defaultRTE">
    <ejs-richtexteditor id="preview" ref="rteInstance"
:toolbarSettings="toolbarSettings" :created="created"
:actionComplete='actionComplete' :editorMode="editorMode" :height="height">
In Rich Text Editor , you click the toolbar buttons to format the words and
the changes
are visible immediately.
Markdown is not like that. When you format the word in Markdown format, you
need to
add Markdown syntax to the word to indicate which words
and phrases should look different from each other
Rich Text Editor supports markdown editing when the editorMode set as
**markdown** and using both *keyboard interaction* and *toolbar action*, you
can apply the formatting to text.
We can add our own custom formation syntax for the Markdown formation,
[sample link](https://ej2.syncfusion.com/home/).
The third-party library <b>Marked</b> is used in this sample to convert
markdown into HTML content.
      </ejs-richtexteditor>
  </div>
</template>
    <style>
    .e-richtexteditor .e-rte-content .e-content{
        min-height: 150px;
    }
    .e-richtexteditor .e-rte-content textarea.e-content {
        float: left;
        border-right: 1px solid rgba(0, 0, 0, 0.12);
    }
    .e-richtexteditor .e-rte-content {
        overflow: hidden;
    }
    .e-md-preview::before {
        content: '\e345';
    }
    .e-rte-content .e-content.e-pre-source {
        width: 100%;
    }
    .e-icon-btn.e-active .e-md-preview.e-icons::before {
        content: '\e350';
    }
</style>
<script>
import Vue from "vue";
import { Browser, addClass, removeClass, isNullOrUndefined } from
"@syncfusion/ej2-base";
import { RichTextEditorPlugin, Toolbar, Link, Image, MarkdownEditor } from
"@syncfusion/ej2-vue-richtexteditor";
import { createElement, KeyboardEventArgs } from '@syncfusion/ej2-vue-base';
import { L10n } from '@syncfusion/ej2-base';
L10n.load({
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    'en-US': {
        'richtexteditor': {
            'TableHeadingText': 'Header',
            'TableColText': 'Cell'
         }
     }
});
Vue.use(RichTextEditorPlugin);
export default {
    data: function() {
        return {
            textArea: '',
            height: '300px',
        toolbarSettings: {
            items: ['Bold', 'Italic', 'StrikeThrough', '|', 'Formats',
'OrderedList', 'UnorderedList', '|', 'CreateLink', 'Image', 'CreateTable',
'|',
                { tooltipText: 'Preview', template: '<button id="preview-
code" class="e-tbar-btn e-control e-btn e-icon-btn">' +
                    '<span class="e-btn-icon e-md-preview e-
icons"></span></button>' },
                { tooltipText: 'Split Editor', template: '<button
id="MD_Preview" class="e-tbar-btn e-control e-btn e-icon-btn">' +
                    '<span class="e-btn-icon e-view-side e-
icons"></span></button>' }, 'FullScreen', '|', 'Undo', 'Redo']
        },
        editorMode: 'Markdown',
        };
    },
    methods: {
        created: function() {
            var script = document.createElement('script');
            script.src =
'https://cdn.jsdelivr.net/npm/marked/marked.min.js';
            document.head.appendChild(script);
            this.textArea =
this.$refs.rteInstance.$el.parentNode.querySelector('.e-content');
            this.textArea.onkeyup = (Event) => {
                this.markDownConversion();
            };
            document.getElementById('preview-code').onclick = () => {
                this.fullPreview({ mode: true, type: 'preview'});
                 if (event.target.parentElement.classList.contains('e-
active')) {

this.$refs.rteInstance.ej2Instances.disableToolbarItem(['Bold', 'Italic',
'StrikeThrough', 'Formats', 'OrderedList',
                    'UnorderedList', 'CreateLink', 'Image',
'CreateTable']);

event.target.parentElement.parentElement.nextElementSibling.classList.add('e
-overlay');
                } else {

this.$refs.rteInstance.ej2Instances.enableToolbarItem(['Bold', 'Italic',
'StrikeThrough', 'Formats', 'OrderedList',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    'UnorderedList', 'CreateLink', 'Image',
'CreateTable']);

event.target.parentElement.parentElement.nextElementSibling.classList.remove
('e-overlay');
                }
            };
            document.getElementById('MD_Preview').onclick = () => {
                if (this.$refs.rteInstance.$el.classList.contains('e-rte-
full-screen')) {
                    this.fullPreview({ mode: true, type: '' });
                }
                document.getElementById('preview-code').classList.remove('e-
active');
                this.$refs.rteInstance.ej2Instances.showFullScreen();
            };
        },
        markDownConversion: function(){
        if (document.getElementById('MD_Preview').classList.contains('e-
active')) {
            var id = this.$refs.rteInstance.ej2Instances.getID() + 'html-
view';
            var htmlPreview =
this.$refs.rteInstance.$el.parentNode.querySelector('#' + id);
            htmlPreview.innerHTML = marked(this.textArea.value);
        }
    },
    actionComplete: function(e) {
        var mdsource = document.getElementById('preview-code');
        var mdSplit = document.getElementById('MD_Preview');
        var id = this.$refs.rteInstance.ej2Instances.getID() + 'html-view';
        var htmlPreview =
this.$refs.rteInstance.$el.parentNode.querySelector('#' + id);
            if (e.targetItem === 'Maximize' && isNullOrUndefined(e.args)) {
                this.fullPreview({ mode: true, type: '' });
            } else if (!mdSplit.parentElement.classList.contains('e-
overlay')) {
                if (e.targetItem === 'Minimize') {
                this.textArea.style.display = 'block';
                this.textArea.style.width = '100%';
                if (htmlPreview) { htmlPreview.style.display = 'none'; }
                mdSplit.classList.remove('e-active');
                mdsource.classList.remove('e-active');
                }
            this.markDownConversion();
            }
        },
        fullPreview: function(event){
            var mdsource = document.getElementById('preview-code');
             var mdSplit = document.getElementById('MD_Preview');
            var id = this.$refs.rteInstance.ej2Instances.getID() + 'html-
view';
            var htmlPreview =
this.$refs.rteInstance.$el.parentNode.querySelector('#' + id);
        if ((mdsource.classList.contains('e-active') ||
mdSplit.classList.contains('e-active')) && event.mode) {
            mdsource.classList.remove('e-active');
```

```
            mdSplit.classList.remove('e-active');
            this.textArea.style.display = 'block';
            this.textArea.style.width = '100%';
            htmlPreview.style.display = 'none';
        } else {
            mdsource.classList.add('e-active');
             mdSplit.classList.add('e-active');
            if (!htmlPreview) {
                htmlPreview = document.createElement('div');
                htmlPreview.id = id;
                htmlPreview.setAttribute('class', 'e-content');
                this.textArea.parentNode.appendChild(htmlPreview);
            }
            if (event.type === 'preview') {
                this.textArea.style.display = 'none';
                htmlPreview.classList.add('e-pre-source');
            } else {
                htmlPreview.classList.remove('e-pre-source');
                this.textArea.style.width = '50%';
            }
            htmlPreview.style.display = 'block';
            htmlPreview.innerHTML =
marked(this.$refs.rteInstance.ej2Instances.contentModule.getEditPanel().valu
e);
            mdsource.parentElement.title = 'Code View';
        }
    }
    },
    provide:{
        richtexteditor:[Toolbar, Link, Image, MarkdownEditor]
    }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs9" %}

Custom formation

The Rich Text Editor allows you to customize the markdown syntax by overriding its default syntax. Configure the customized markdown syntax using the formatterproperty.

This sample demonstrates how to customize tags of markdown formatting.

Copyright © 2001 -2024 Syncfusion Inc.

For example, apply + to Unordered list, apply 1., 2., 3. to Ordered list, for bold, , and for italic _.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="rteInstance" height="350" editorMode="Markdown"
:value="value" :toolbarSettings="toolbarConfig" :created="created"
:formatter="formatter">
  </ejs-richtexteditor>
</template>
<style>
  .e-richtexteditor .e-rte-content .e-content{
        min-height: 150px;
    }
    .e-richtexteditor .e-rte-content textarea.e-content {
        float: left;
        border-right: 1px solid rgba(0, 0, 0, 0.12);
    }
    .e-richtexteditor .e-rte-content {
        overflow: hidden;
    }
    .e-md-preview::before {
        content: '\e345';
    }
    .e-rte-content .e-content.e-pre-source {
        width: 100%;
    }
    .e-icon-btn.e-active .e-md-preview.e-icons::before {
        content: '\e350';
    } content: '\e350';
  }
</style>
<script>
  import Vue from 'vue';
  import { createElement, KeyboardEventArgs } from "@syncfusion/ej2-base";
  import { RichTextEditorPlugin, Toolbar, Link, Image, MarkdownEditor,
MarkdownFormatter } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        value: `***Overview***
  The Rich Text Editor component is WYSIWYG ("what you see is what you get")
editor used to create and edit the content and return valid HTML markup or
markdown (MD) of the content. The editor provides a standard toolbar to
format content using its commands. Modular library features to load the
necessary functionality on demand. The toolbar contains commands to align
the text, insert link, insert image, insert list, undo/redo operation, HTML
view, and more.
  ***Key features***
  - *Mode*: Provides IFRAME and DIV mode.
  - *Module*: Modular library to load the necessary functionality on demand.
  - *Toolbar*: Provide a fully customizable toolbar.
  - *Editing*: HTML view to edit the source directly for developers.
  - *Third-party Integration*: Supports to integrate third-party library.
  - *Preview*: Preview the modified content before saving it.
  - *Tools*: Handling images, hyperlinks, video, uploads and more.
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  - *Undo and Redo*: Undo/redo manager.
  - *Lists*:Creates bulleted and numbered list.`,
        toolbarConfig: {
          items: [
            'Bold', 'Italic', 'StrikeThrough', '|',
            'Formats', 'OrderedList', 'UnorderedList', '|',
            'CreateLink', 'Image', '|',
            {
              tooltipText: 'Preview',
              template: '<button id="preview-code" class="e-tbar-btn e-
control e-btn e-icon-btn">' +
                '<span class="e-btn-icon e-md-preview e-
icons"></span></button>'
            },
            '|', 'Undo', 'Redo'
          ]
        },
        formatter: new MarkdownFormatter({
            listTags: { 'OL': '1., 2., 3.', 'UL': '+ ' },
            formatTags: {
                'Blockquote': '> '
            },
            selectionTags: {'Bold': ' __ ',  'Italic': ' _ '}
        }),
      }
    },
    methods: {
      created: function() {
        var script = document.createElement('script');
        script.src = 'https://cdn.jsdelivr.net/npm/marked/marked.min.js';
        document.head.appendChild(script);
        this.textArea =
this.$refs.rteInstance.$el.parentNode.querySelector('.e-content');
        this.textArea.onkeyup = (Event) => {
        this.markDownConversion();
        };
        document.getElementById('preview-code').onclick = () => {
          this.fullPreview({ mode: true, type: 'preview'});
          if (event.target.parentElement.classList.contains('e-active')) {
            this.$refs.rteInstance.ej2Instances.disableToolbarItem([
              'Bold', 'Italic', 'StrikeThrough', 'Formats', 'OrderedList',
              'UnorderedList', 'CreateLink', 'Image'
            ]);

event.target.parentElement.parentElement.nextElementSibling.classList.add('e
-overlay');
          } else {
            this.$refs.rteInstance.ej2Instances.enableToolbarItem([
              'Bold', 'Italic', 'StrikeThrough', 'Formats', 'OrderedList',
              'UnorderedList', 'CreateLink', 'Image'
            ]);

event.target.parentElement.parentElement.nextElementSibling.classList.remove
('e-overlay');
          }
        };
      },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      markDownConversion: function() {
        if (document.getElementById('MD_Preview').classList.contains('e-
active')) {
          var id = this.$refs.rteInstance.ej2Instances.getID() + 'html-
view';
          var htmlPreview = this.$refs.rteInstance.$el.querySelector('#' +
id);
          htmlPreview.innerHTML = marked(this.textArea.value);
        }
      },
      fullPreview: function(event){
        var mdSource = document.getElementById('preview-code');
        var id = this.$refs.rteInstance.ej2Instances.getID() + 'html-view';
        var htmlPreview =
this.$refs.rteInstance.$el.parentNode.querySelector('#' + id);
        if ((mdSource.classList.contains('e-active')) && event.mode) {
          mdSource.classList.remove('e-active');
          this.textArea.style.display = 'block';
          this.textArea.style.width = '100%';
          htmlPreview.style.display = 'none';
        } else {
          mdSource.classList.add('e-active');
          if (!htmlPreview) {
            htmlPreview = document.createElement('div');
            htmlPreview.id = id;
            htmlPreview.setAttribute('class', 'e-content');
            this.textArea.parentNode.appendChild(htmlPreview);
          }
          if (event.type === 'preview') {
            this.textArea.style.display = 'none';
            htmlPreview.classList.add('e-pre-source');
          } else {
            htmlPreview.classList.remove('e-pre-source');
            this.textArea.style.width = '50%';
          }
          htmlPreview.style.display = 'block';
          htmlPreview.innerHTML =
marked(this.$refs.rteInstance.ej2Instances.contentModule.getEditPanel().valu
e);
          mdSource.parentElement.title = 'Code View';
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, MarkdownEditor]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs10" %}

See Also

- How to integrate the third party library
- How to change the editor mode

File browser in Vue Rich text editor component

Rich Text Editor allows to browse and insert images in the edit panel using the file browser. File browser allows the users to browse and select a file or folder from the file system and it supports various cloud services.

Required additional dependency

The following list of additional dependencies are required to use the file browser feature in the Rich Text Editor.

`js

|-- @syncfusion/ej2-vue-richtexteditor

|-- @syncfusion/ej2-layouts

|-- @syncfusion/ej2-grids

|-- @syncfusion/ej2-filemanager

`

Additional CSS Reference

Additionally add the styles as given below in `<style>` section of the `App.vue` file.

`

<style>

@import "../node_modules/@syncfusion/ej2-layouts/styles/material.css";

@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";

@import "../node_modules/@syncfusion/ej2-filemanager/styles/material.css";

</style>

`

The following example explains about how to configure the file browser within the Rich Text Editor component.

- Configure the `FileManager` toolbar item in the `toolbarSettings` API `items` property.

Copyright © 2001 -2024 Syncfusion Inc.

- Set enable property as true on fileManagerSettings property to make the file browser in the Rich Text Editor to appear on the FileManager toolbar click action.

Rich Text Editor features are segregated into individual feature-wise modules. To use the file browser tool, configure FileManager in the provider.

APP.VUE

```
<template>
  <ejs-richtexteditor :toolbarSettings="toolbarSettingsData"
:fileManagerSettings="fileManagerSettingsData">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar, FileManager } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        toolbarSettingsData: {
          items: ['FileManager']
        },
```

```
        fileManagerSettingsData: {
          enable: true,
          path: '/Pictures/Food',
          ajaxSettings: {
              url: 'https://ej2-aspcore-
service.azurewebsites.net/api/FileManager/FileOperations',
              getImageUrl: 'https://ej2-aspcore-
service.azurewebsites.net/api/FileManager/GetImage',
              uploadUrl: 'https://ej2-aspcore-
service.azurewebsites.net/api/FileManager/Upload',
              downloadUrl: 'https://ej2-aspcore-
service.azurewebsites.net/api/FileManager/Download'
          }
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar,
FileManager]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-layouts/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
filemanager/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/file-browser-cs1" %}

Format Painter in Vue Rich Text Editor Component | Syncfusion

A format painter is a tool that allows you to copy the formatting from a piece of text and apply it to another one. Format Painter can be accessed via the toolbar or the keyboard shortcuts. The format painter can copy the formatting of a single word or a whole paragraph. The format painter can be customized using the formatPainterSettings property.

Enabling the toolbar option for Format Painter

You can add the `FormatPainter` tool in the Rich Text Editor using the `toolbarSettings` items property.

Rich Text Editor features are segregated into individual feature-wise modules. To use the Format Painter feature, we need to import and inject the `FormatPainter` module in `provides` section.

Copyright © 2001 -2024 Syncfusion Inc.

By double-clicking the format painter toolbar button, sticky mode will be enabled. In sticky mode, the format painter will be disabled when the user clicks the Escape key again.

The following code example shows how to add the format painter tool in the Rich Text Editor.

APP.VUE

```
<template>
    <div>
        <div class="control-section">
            <div class="sample-container">
                <div class="default-section">
                <ejs-richtexteditor ref="rteObj"
:toolbarSettings="toolbarSettings" >
                    <h3><strong>Format Painter</strong></h3>
                    <p>
                        A Format Painter is a Rich Text Editor feature
allowing users to quickly
                        <span style="background-color: rgb(198, 140,
83);"><strong>copy</strong></span>
                        and
                        <span style="background-color: rgb(198, 140,
83);"><strong>paste</strong></span>
                        formatting from one text to another. With a rich
text editor, utilize the format painter as follows:
                    </p>
                    <ul>
                        <li>
                            Select the text whose format you want to copy.
                        </li>
                        <li>
                            Click on the <strong><em>Format
Painter</em></strong> button in the toolbar. It may look like a paintbrush
icon.
                        </li>
                        <li>
                            The cursor will change to a
<strong>paintbrush</strong> icon. Click and drag the cursor over the text
you want to apply the copied format.
                        </li>
                        <li>
                            Release the mouse button to apply the format.
                        </li>
                    </ul>
                    <p>
                        Using the format painter in a rich text editor can
save you time when formatting a large document, You can quickly
                        copy and apply formatting
                        to <span style="background-color: rgb(198, 140,
83);"><strong>multiple sections</strong></span>.
                        It's a helpful tool for anyone who works with text
editing regularly, such as writers, editors, and content creators.
                    </p>
                </ejs-richtexteditor>
                </div>
            </div>
        </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
</template>
    <script>
    import Vue from "vue";
    import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar, FormatPainter } from "@syncfusion/ej2-vue-richtexteditor";

    Vue.use(RichTextEditorPlugin);

    export default {
         data: function() {
            return {
                toolbarSettings: {
                    items: ['FormatPainter', 'ClearFormat', 'Bold',
'Italic', 'Underline', '|', 'Formats', 'Alignments',
                    'OrderedList', 'UnorderedList', '|', 'CreateLink',
'Image', '|', 'SourceCode', 'Undo', 'Redo']
                }
            };
        },
        provide:{
            richtexteditor:[Toolbar, Link, Image, HtmlEditor, QuickToolbar,
FormatPainter]
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/format-painter-cs1" %}

Customization of copy and paste format

You can customize the format painter tool in the Rich Text Editor using the formatPainterSettings property.

The allowedFormats property helps you to specify tag names that allow the formats to be copied from the selected text. For instance, you can include formats from the selected text using tags like p; h1; h2; h3; div; ul; ol; li; span; strong; em; code;. The following example demonstrates how to customize this functionality.

Similarly, with the deniedFormats property, you can utilize the selectors to prevent specific formats from being pasted onto the selected text. The table below illustrates the selectors and their respective usage.

Type	Description	Selector	Usage

Copyright © 2001 -2024 Syncfusion Inc.

|------|-------------------|---|---|

| () | Class Selector | h3(e-rte-block-blue-text) | The class name e-rte-block-blue-text of H3 element is not copied. |

| [] | Attribute Selector | span\[title] | The title attribute of span element is not copied. |

| {} | Style Selector | span{background-color, color} | The background-color and color styles of span element is not copied. |

Using the deniedFormats property following styles are denied copying from the selected text such as h3(e-rte-block-blue-text){background-color,padding}[title]; li{color}; span(e-inline-text-highlight)[title]; strong{color}(e-rte-strong-bg).

APP.VUE

```
<template>
    <div>
        <div class="control-section">
            <div class="sample-container">
                <div class="default-section">
                <ejs-richtexteditor ref="rteObj"
:formatPainterSettings="formatPainterSettings"
:toolbarSettings="toolbarSettings" >
                    <h3 class="e-rte-block-blue-text" title="Format Painter"
style="color: #0079f3; background-color: #eff6ff; padding: 10px;"><strong
>Format Painter</strong></h3>
                    <p>
                        A Format Painter is a Rich Text Editor feature
allowing users to quickly
                        <span class="e-inline-text-highlight" style="color:
blue;" title="Styled by CSS Class selector"><strong>copy</strong></span>
                        and
                        <span class="e-inline-text-highlight" style="color:
blue;"  title="Styled by CSS Class selector"><strong>paste</strong></span>
                        formatting from one text to another. With a rich
text editor, utilize the format painter as follows:
                    </p>
                    <ul>
                        <li style="color: crimson;">
                            Select the text whose format you want to copy.
                        </li>
                        <li style="color: crimson;">
                            Click on the <strong><em>Format
Painter</em></strong> button in the toolbar. It may look like a paintbrush
icon.
                        </li>
                        <li style="color: crimson;">
                            The cursor will change to a
<strong>paintbrush</strong> icon. Click and drag the cursor over the text
you want to apply the copied format.
                        </li>
                        <li style="color: crimson;">
                            Release the mouse button to apply the format.
                        </li>
```

```
                    </ul>
                    <p>
                        Using the format painter in a rich text editor can
save you time when formatting a large document, You can quickly
                        copy and apply formatting
                        to <strong class="e-rte-strong-bg" style="color:
blue">multiple sections</strong>.
                        It's a helpful tool for anyone who works with text
editing regularly, such as writers, editors, and content creators.
                    </p>
                </ejs-richtexteditor>
                </div>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar, FormatPainter } from "@syncfusion/ej2-vue-richtexteditor";
    Vue.use(RichTextEditorPlugin);
    export default {
        data: function() {
            return {
                toolbarSettings: {
                    items: ['FormatPainter', 'ClearFormat', 'Bold',
'Italic', 'Underline', '|', 'Formats', 'Alignments',
                    'OrderedList', 'UnorderedList', '|', 'CreateLink',
'Image', '|', 'SourceCode', 'Undo', 'Redo']
                },
                formatPainterSettings: {
                    allowedFormats:
'p;h1;h2;h3;div;ul;ol;li;span;strong;em;code;',
                    deniedFormats: 'h3(e-rte-block-blue-text){background-
color,padding,color}[title]; li{color}; span(e-inline-text-
highlight){color}[title]; strong{color}(e-rte-strong-bg);',
                }
            };
        },
        provide:{
            richtexteditor:[Toolbar, Link, Image, HtmlEditor, QuickToolbar,
FormatPainter]
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/rich-text-editor/format-painter-cs2" %}

Using the shortcut key to copy and paste the format

You can use the following shortcut keys to copy and paste the format in the Rich Text Editor.

Actions	Keyboard shortcuts	Description
Copy the format	Alt + Shift + c	Copy the selection format or current range.
Pate the format	Alt + Shift + v	Paint the copied format.
Escape	Esc	Remove the previously copied format and disable the sticky mode.

The format painter retains the formatting after application making it possible to apply the same formatting multiple times by using the Alt + Shift + v keyboard shortcut.

Additionally, You can perform the format painter actions programmatically using the executeCommand public method.

Form support in Vue Rich text editor component

The following sample demonstrates how to get the Rich Text Editor value in button click.

Render the Rich Text Editor

Render the Rich Text Editor in form.

APP.VUE

```
<template>
<div id="app">
    <div id='container'>
        <br>
        <form id="myForm" >
          <ejs-richtexteditor name="name" id="name" v-model="name"
:blur="blur"></ejs-richtexteditor>
          <span style='color: red' v-for="error in errors">{{ error
}}</span>
          <p>
    <button id="validateSubmit" class="sample-btn e-control e-btn"
type="submit" data-ripple="true">Submit</button>
          <button id="reset-btn" class="sample-btn e-control e-btn"
type="reset" data-ripple="true">Reset</button>
  </p>
      </form>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor } from
"@syncfusion/ej2-vue-richtexteditor";
import { FormValidator } from '@syncfusion/ej2-inputs';
Vue.use(RichTextEditorPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data:{
    errors:[],
    name:null,
  },
  methods:{
    blur: function(){
      if(this.name) return true;
      this.errors = [];
      if(!this.name) this.errors.push("value required.");
    }
  },
  provide:{
        richtexteditor:[Toolbar, Link, Image, HtmlEditor]
    }
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs2" %}

See Also

- How to integrate the third party library
- How to validate the value

Validation in Vue Rich text editor component

Validation rules

The Rich Text Editor provides the functionality of character count and its validation. So, you can validate the Rich Text Editor's value on form submission by applying validationRules and validationMessage to the Rich Text Editor.

| Rules | Description |

|---|---|

| required | Requires value for the Rich Text Editor control.|

| minlength | Requires the value to be of given minimum characters count.|

| maxlength | Requires the value to be of given maximum characters count.|

Copyright © 2001 -2024 Syncfusion Inc.

This sample is used to validate form using the obtrusive Validation. Type the values in Rich Text Editor and the form enables the validation with the formvalidator rules by clicking on the submit externally. All rules are validated by the formvalidator rules.

APP.VUE

```
<template>
<div id="app">
    <div id='container'>
        <br>
        <form id="myForm" class="form-vertical">
        <div class="form-group">
          <ejs-richtexteditor id="defaultRTE"  name="defaultRTE"
class="form-control" :placeholder="placeholder"
:showCharCount="showCharCount" :created="forms" :change="onChange"
:maxLength="maxLength"></ejs-richtexteditor>
          <div class="error-element"></div>
        </div>
        <br/>
        <div class="form-btn-section">
          <ejs-button ref='submitButtonRef' :disabled="true"
id="validateSubmit"  >Submit</ejs-button>
          <button id="reset-btn" class="sample-btn e-control e-btn"
type="reset" data-ripple="true">Reset</button>
        </div>
      </form>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Count, Image, HtmlEditor }
from "@syncfusion/ej2-vue-richtexteditor";
import { FormValidator } from '@syncfusion/ej2-inputs';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(RichTextEditorPlugin);
Vue.use(ButtonPlugin);
export default {
   data: function() {
    return {
      value : null,
      showCharCount: true,
      maxLength: 100,
      placeholder: 'Type something'
        }
    },
methods: {
     forms() {
      var option = {
        rules: {
          defaultRTE: {
            required: true
          }
        },
        customPlacement: function(element, errorElement) {
          document.querySelector(".error-
element").appendChild(errorElement);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        }
      }
      var formObject = new FormValidator('#myForm', option);
     },
     onChange() {
      this.$refs.submitButtonRef.disabled =false;
     }
   },
  provide:{
        richtexteditor:[Toolbar, Link, Image, HtmlEditor, Count]
    }
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
.error-element {
  margin-top: 5px;
}
.sample-btn {
  margin-left : 10px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs12" %}

Validation message

The default error message for a rule can be customizable by defining it along with the concern rule object as follows.

In the following sample, customize the error message along with the concern rule.

APP.VUE

```
<template>
<div id="app">
    <div id='container'>
        <br>
        <form id="myForm" class="form-vertical">
        <div class="form-group">
          <ejs-richtexteditor id="defaultRTE"  name="defaultRTE"
class="form-control" :placeholder="placeholder"
:showCharCount="showCharCount" :created="forms" :change="onChange"
:maxLength="maxLength"></ejs-richtexteditor>
          <div class="error-element"></div>
        </div>
        <br/>
```

```
        <div class="form-btn-section">
         <ejs-button ref='submitButtonRef' :disabled="true"
id="validateSubmit"  >Submit</ejs-button>
          <button id="reset-btn" class="sample-btn e-control e-btn"
type="reset" data-ripple="true">Reset</button>
        </div>
      </form>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Image, Count, HtmlEditor }
from "@syncfusion/ej2-vue-richtexteditor";
import { FormValidator } from '@syncfusion/ej2-inputs';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
 Vue.use(RichTextEditorPlugin);
 Vue.use(ButtonPlugin);
 export default {
   data: function() {
    return {
      value :null,
      showCharCount: true,
      maxLength: 100,
      placeholder: 'Type something'
        }
        },
methods: {
     forms() {
      var option = {
        rules: {
          defaultRTE: {
            required: true,
            minLength: [15, 'Need atleast 8 character length'],
            maxLength:[1208, 'Maximum 100 character only']
          }
        },
        customPlacement: function(element, errorElement) {
          document.querySelector(".error-
element").appendChild(errorElement);
        }
      }
      var formObject = new FormValidator('#myForm', option);
     },
     onChange() {
        this.$refs.submitButtonRef.disabled =false;
     }
   },
  provide:{
        richtexteditor:[Toolbar, Link, Image, HtmlEditor, Count]
    }
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
```

```
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
.error-element {
  margin-top: 5px;
}
.sample-btn {
  margin-left : 10px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs13" %}

Globalization in Vue Rich text editor component

Localization

The Rich Text Editor provides an option to localize its strings; it is used to adapting the editor to a particular local language. By default, the editor will use the US English (en-US) as its language. Please find the table with a list of keys and their corresponding values for the default language (en-US).

` javascript

'en-US': {

'richtexteditor': {

"alignments": "Alignments",

"justifyleft": "JustifyLeft",

"justifycenter": "JustifyCenter",

"justifyright": "Justify Right",

"justifyfull": "Justify Full",

"formats": "Formats",

"fontname": "Font Name",

"fontsize": "Font Size",

"fontcolor": "Font Color",

"backgroundcolor": "Background color",

"bold": "Bold",

"italic": "Italic",

"underline": "Underline",

"strikethrough": "Strikethrough",

"clearall": "Clear All",

Copyright © 2001 -2024 Syncfusion Inc.

"clearformat": "Clear Format",
"cut": "Cut",
"copy": "Copy",
"paste": "Paste",
"unorderedlist": "unordered list",
"orderedlist": "ordered list",
"indent": "Increase Indent",
"outdent": "Decrease Indent",
"undo": "Undo",
"redo": "Redo",
"superscript": "Superscript",
"subscript": "Subscript",
"createlink": "Insert link",
"removelink": "remove link",
"openlink": "Open link",
"editlink": "Edit link",
"image": "Insert image",
"replace": "Replace",
"align": "Align",
"caption": "Caption",
"remove": "Remove",
"insertlink": "insertlink",
"display": "Display",
"alttext": "Alternative Text",
"dimension": "Dimension",
"fullscreen": "Full Screen",
"maximize": "Maximize",
"minimize": "Minimize",
"zoomin": "Zoom In",
"zoomout": "Zoom Out",
"uppercase": "Upper Case",
"lowercase": "Lower Case",
"print": "Print",

Copyright © 2001 -2024 Syncfusion Inc.

"sourcecode": "Source Code",
"preview": "Preview",
"viewside": "View Side",
"insertcode": "Insert Code",
"linkText": "Display Text",
"linkTooltipLabel": "Title",
"linkWebUrl": "Web Address",
"linkTitle": "Enter a title",
"linkOpenInNewWindow": "Open Link in New Window",
"linkHeader": "Insert Link",
"dialogInsert": "Insert",
"dialogCancel": "Cancel",
"dialogUpdate": "Update",
"imageHeader": "Insert Image",
"imageLinkHeader": "You can also provide a link from the web",
"mdimageLink": "Please provide a URL for your image",
"imageUploadMessage": "Drop image here or browse to upload",
"imageDeviceUploadMessage": "Click here to upload",
"imageAlternateText": "Alternate Text",
"alternateHeader": "Alternative Text",
"browse": "Browse",
"imageCaption": "Caption",
"imageSizeHeader": "Image Size",
"imageHeight": "Height",
"imageWidth": "Width",
"textPlaceholder": "Enter Text",
"inserttablebtn": "Insert Table",
"tabledialogHeader": "Insert Table",
"tableWidth": "Width",
"cellpadding": "Cell Padding",
"cellspacing": "Cell Spacing",
"columns": "Number of columns",
"rows": "Number of rows",

Copyright © 2001 -2024 Syncfusion Inc.

"tableRows": "Table Rows",
"tableColumns": "Table Columns",
"tableCellHorizontalAlign": "Table Cell Horizontal Align",
"tableCellVerticalAlign": "Table Cell Vertical Align",
"createTable": "Create Table",
"removeTable": "Remove Table",
"tableHeader": "Table Header",
"tableRemove": "Table Remove",
"tableCellBackground": "Table Cell Background",
"tableEditProperties": "Table Edit Properties",
"styles": "Styles",
"insertColumnLeft": "Insert Column Left",
"insertColumnRight": "Insert Column Right",
"deleteColumn": "Delete Column",
"insertRowBefore": "Insert Row Before",
"insertRowAfter": "Insert Row After",
"deleteRow": "Delete Row",
"tableEditHeader": "Edit Table",
"TableHeadingText": "Heading",
"TableColText": "Col",
"imageInsertLinkHeader": "Insert Link",
"editImageHeader": "Edit Image",
"alignmentsDropDownLeft": "Align Left",
"alignmentsDropDownCenter": "Align Center",
"alignmentsDropDownRight": "Align Right",
"alignmentsDropDownJustify": "Align Justify",
"imageDisplayDropDownInline": "Inline",
"imageDisplayDropDownBreak": "Break",
"tableInsertRowDropDownBefore": "Insert row before",
"tableInsertRowDropDownAfter": "Insert row after",
"tableInsertRowDropDownDelete": "Delete row",
"tableInsertColumnDropDownLeft": "Insert column left",
"tableInsertColumnDropDownRight": "Insert column right",

Copyright © 2001 -2024 Syncfusion Inc.

"tableInsertColumnDropDownDelete": "Delete column",
"tableVerticalAlignDropDownTop": "Align Top",
"tableVerticalAlignDropDownMiddle": "Align Middle",
"tableVerticalAlignDropDownBottom": "Align Bottom",
"tableStylesDropDownDashedBorder": "Dashed Borders",
"tableStylesDropDownAlternateRows": "Alternate Rows",
"pasteFormat": "Paste Format",
"pasteFormatContent": "Choose the formatting action",
"plainText": "Plain Text",
"cleanFormat": "Clean",
"keepFormat": "Keep",
"formatsDropDownParagraph": "Paragraph",
"formatsDropDownCode": "Code",
"formatsDropDownQuotation": "Quotation",
"formatsDropDownHeading1": "Heading 1",
"formatsDropDownHeading2": "Heading 2",
"formatsDropDownHeading3": "Heading 3",
"formatsDropDownHeading4": "Heading 4",
"fontNameSegoeUI": "Segoe UI",
"fontNameArial": "Arial",
"fontNameGeorgia": "Georgia",
"fontNameImpact": "Impact",
"fontNameTahoma": "Tahoma",
"fontNameTimesNewRoman": "Times New Roman",
"fontNameVerdana": "Verdana",
"numberFormatListNumber": 'Number',
"numberFormatListLowerAlpha": 'LowerAlpha',
"numberFormatListUpperAlpha": 'UpperAlpha',
"numberFormatListLowerRoman": 'LowerRoman',
"numberFormatListUpperRoman": 'UpperRoman',
"numberFormatListLowerGreek": 'LowerGreek',
"bulletFormatListDisc": 'Disc',
"bulletFormatListCircle": 'Circle',

Copyright © 2001 -2024 Syncfusion Inc.

"bulletFormatListSquare": 'Square',

"numberFormatListNone": 'None',

"bulletFormatListNone":'None',

"formatPainter": 'Format Painter',

"emojiPicker": 'Emoji Picker',

"embeddedCode": 'Embedded Code',

"pasteEmbeddedCodeHere": 'Paste Embedded Code here',

"emojiPickerTypeToFind": 'Type to find',

"emojiPickerNoResultFound": 'No results found',

"emojiPickerTrySomethingElse": 'Try something else',

}

}

`

To localize the editor's strings with your own localization, copy the default language informations and localize the strings in the values column. For example, to localize the editor in German language ("de-DE").

` javascript

'de-DE': {

'richtexteditor': {

"alignments": "Alignments",

"justifyLeft": "Ausrichten von Text links",

"justifyCenter": "Text-Zentrum",

"justifyRight": "Ausrichten von Text rechts",

"justifyFull": "rechtfertigen",

"fontName": "Wählen Sie Schriftfamilie",

"fontSize": "Wählen Sie Schriftgröße",

"fontColor": "Wählen Sie die Farbe",

"backgroundColor": "Hintergrundfarbe",

"bold": "fett",

"italic": "kursiv",

"underline": "unterstreichen",

"strikethrough": "Durchgestrichen",

"clearAll": "Alles",

"clearFormat": "Klar Format",

Copyright © 2001 -2024 Syncfusion Inc.

"cut": "schneiden",
"copy": "Kopieren",
"paste": "Paste",
"unorderedList": "Legen Sie ungeordnete Liste",
"orderedList": "Geordnete Liste einfügen",
"indent": "Einzug",
"outdent": "Einzug verkleinern",
"undo": "lösen",
"redo": "Wiederherstellen",
"superscript": "Überschrift",
"subscript": "index",
"createLink": "Link einfügen",
"removeLink": "fjern Hyperlink",
"openLink": "Open link",
"editLink": "Edit link",
"image": "Bild einfügen",
"replace": "ersetzen",
"align": "ausrichten",
"caption": "Bildbeschriftung",
"formats": "Formats",
"remove": "Löschen",
"insertLink": "Link einfügen",
"display": "Anzeige",
"alttext": "alternativer Text",
"dimension": "Größe",
"fullscreen": "Vollbild",
"maximize": "Maximieren",
"minimize": "minimieren",
"zoomIn": "hineinzoomen",
"zoomOut": "Rauszoomen",
"upperCase": "Großbuchstaben",
"lowerCase": "Kleinbuchstaben",
"print": "Drucken",

Copyright © 2001 -2024 Syncfusion Inc.

"sourcecode": "Quellcode",
"preview": "Vorschau",
"viewside": "Seite anzeigen",
"insertcode": "Code eingeben",
"linkText": "Displaytekst",
"linkTooltipLabel": "tooltip",
"linkWebUrl": "Webadres",
"linkOpenInNewWindow": "Open de link in een nieuw venster",
"linkHeader": "Link invoegen",
"dialogInsert": "invoegen",
"dialogCancel": "Annuleer",
"dialogUpdate": "Bijwerken",
"imageHeader": "Voeg afbeelding in",
"imageLinkHeader": "U kunt ook een link van internet opgeven",
"imageUploadMessage": "Zet hier een afbeelding neer of klik om te uploaden",
"imageDeviceUploadMessage": "Klik hier om te uploaden",
"imageAlternateText": "Alternatieve tekst",
"alternateHeader": "Alternatieve tekst",
"browse": "Blader",
"imageUrl": "URL",
"imageCaption": "onderschrift",
"imageSizeHeader": "Afbeeldingsgrootte",
"imageHeight": "Hoogte",
"imageWidth": "Breedte",
"textPlaceholder": "Text eingeben",
"inserttablebtn": "Tabelle einfügen",
"tabledialogHeader": "Tabelle einfügen",
"tableWidth": "Breite",
"cellpadding": "Zellauffüllung",
"cellspacing": "Zellabstand",
"columns": "Anzahl der Spalten",
"rows": "Reihenanzahl",
"tableRows": "Tabellenzeilen",

Copyright © 2001 -2024 Syncfusion Inc.

"tableColumns": "Tabellenspalten",

"tableCellHorizontalAlign": "Horizontale Ausrichtung der Tabellenzelle",

"tableCellVerticalAlign": "Vertikale Ausrichtung der Tabellenzelle",

"createTable": "Tabelle erstellen",

"removeTable": "Tabelle entfernen",

"tableHeader": "Tabellenkopfzeile",

"tableRemove": "Tabelle entfernen",

"tableCellBackground": "Tabellenzellenhintergrund",

"tableEditProperties": "Eigenschaften der Tabellenbearbeitung",

"styles": "Styles",

"insertColumnLeft": "Spalte links einfügen",

"insertColumnRight": "Spalte rechts einfügen",

"deleteColumn": "Spalte löschen",

"insertRowBefore": "Zeile vor einfügen",

"insertRowAfter": "Zeile einfügen nach",

"deleteRow": "Zeile löschen",

"tableEditHeader": "Tabelle bearbeiten",

"TableHeadingText": "Überschrift",

"TableColText": "Col",

"imageInsertLinkHeader": "Link einfügen",

"editImageHeader": "Bild bearbeiten",

"alignmentsDropDownLeft": "Linksbündig",

"alignmentsDropDownCenter": "Im Zentrum anordnen",

"alignmentsDropDownRight": "Rechts ausrichten",

"alignmentsDropDownJustify": "Justize ausrichten",

"imageDisplayDropDownInline": "In der Reihe",

"imageDisplayDropDownBreak": "Brechen",

"tableInsertRowDropDownBefore": "Reihe vorher einfügen",

"tableInsertRowDropDownAfter": "Zeile danach einfügen",

"tableInsertRowDropDownDelete": "Zeile löschen",

"tableInsertColumnDropDownLeft": "Spalte links einfügen",

"tableInsertColumnDropDownRight": "Spalte rechts einfügen",

"tableInsertColumnDropDownDelete": "Spalte löschen",

Copyright © 2001 -2024 Syncfusion Inc.

"tableVerticalAlignDropDownTop": "Top ausrichten",

"tableVerticalAlignDropDownMiddle": "Mitte ausrichten",

"tableVerticalAlignDropDownBottom": "Unten ausrichten",

"tableStylesDropDownDashedBorder": "Gestrichelte Grenzen",

"tableStylesDropDownAlternateRows": "Alternative Zeilen",

"pasteFormat": "Format einfügen",

"pasteFormatContent": "Wählen Sie die Formatierungsaktion aus",

"plainText": "Einfacher Text",

"cleanFormat": "sauber",

"keepFormat": "Behalten",

"formatsDropDownParagraph": "Absatz",

"formatsDropDownCode": "Kodex",

"formatsDropDownQuotation": "Zitat",

"formatsDropDownHeading1": "Überschrift 1",

"formatsDropDownHeading2": "Überschrift 2",

"formatsDropDownHeading3": "Überschrift 3",

"formatsDropDownHeading4": "Überschrift 4",

"fontNameSegoeUI": "Segoe UI",

"fontNameArial": "Arial",

"fontNameGeorgia": "Georgia",

"fontNameImpact": "Einschlag'",

"fontNameTahoma": "Tahoma",

"fontNameTimesNewRoman": "Mal Neu römisch",

"fontNameVerdana": "Verdana"

}

}

`

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340" locale="de-DE">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
```

```
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { L10n } from '@syncfusion/ej2-base';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  L10n.load({
    'de-DE': {
      'richtexteditor': {
        "alignments": "Alignments",
        "justifyLeft": "Ausrichten von Text links",
        "justifyCenter": "Text-Zentrum",
        "justifyRight": "Ausrichten von Text rechts",
        "justifyFull": "rechtfertigen",
        "fontName": "Wählen Sie Schriftfamilie",
        "fontSize": "Wählen Sie Schriftgröße",
        "fontColor": "Wählen Sie die Farbe",
        "backgroundColor": "Hintergrundfarbe",
        "bold": "fett",
        "italic": "kursiv",
        "underline": "unterstreichen",
        "strikethrough": "Durchgestrichen",
        "clearAll": "Alles",
        "clearFormat": "Klar Format",
        "cut": "schneiden",
        "copy": "Kopieren",
        "paste": "Paste",
        "unorderedList": "Legen Sie ungeordnete Liste",
        "orderedList": "Geordnete Liste einfügen",
        "indent": "Einzug",
        "outdent": "Einzug verkleinern",
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        "undo": "lösen",
        "redo": "Wiederherstellen",
        "superscript": "Überschrift",
        "subscript": "index",
        "createLink": "Link einfügen",
        "removeLink": "fjern Hyperlink",
        "openLink": "Open link",
        "editLink": "Edit link",
        "image": "Bild einfügen",
        "replace": "ersetzen",
        "align": "ausrichten",
        "caption": "Bildbeschriftung",
        "formats": "Formats",
        "remove": "Löschen",
        "insertLink": "Link einfügen",
        "display": "Anzeige",
        "alttext": "alternativer Text",
        "dimension": "Größe",
        "fullscreen": "Vollbild",
        "maximize": "Maximieren",
        "minimize": "minimieren",
        "zoomIn": "hineinzoomen",
        "zoomOut": "Rauszoomen",
        "upperCase": "Großbuchstaben",
        "lowerCase": "Kleinbuchstaben",
        "print": "Drucken",
        "sourcecode": "Quellcode",
        "preview": "Vorschau",
        "viewside": "Seite anzeigen",
        "insertcode": "Code eingeben",
        "linkText": "Displaytekst",
        "linkTooltipLabel": "tooltip",
        "linkWebUrl": "Webadres",
        "linkOpenInNewWindow": "Open de link in een nieuw venster",
        "linkHeader": "Link invoegen",
        "dialogInsert": "invoegen",
        "dialogCancel": "Annuleer",
        "dialogUpdate": "Bijwerken",
        "imageHeader": "Voeg afbeelding in",
        "imageLinkHeader": "U kunt ook een link van internet opgeven",
        "imageUploadMessage": "Zet hier een afbeelding neer of klik om te
uploaden",
        "imageDeviceUploadMessage": "Klik hier om te uploaden",
        "imageAlternateText": "Alternatieve tekst",
        "alternateHeader": "Alternatieve tekst",
        "browse": "Blader",
        "imageUrl": "URL",
        "imageCaption": "onderschrift",
        "imageSizeHeader": "Afbeeldingsgrootte",
        "imageHeight": "Hoogte",
        "imageWidth": "Breedte",
        "textPlaceholder": "Text eingeben",
        "inserttablebtn": "Tabelle einfügen",
        "tabledialogHeader": "Tabelle einfügen",
        "tableWidth": "Breite",
        "cellpadding": "Zellauffüllung",
        "cellspacing": "Zellabstand",
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        "columns": "Anzahl der Spalten",
        "rows": "Reihenanzahl",
        "tableRows": "Tabellenzeilen",
        "tableColumns": "Tabellenspalten",
        "tableCellHorizontalAlign": "Horizontale Ausrichtung der
Tabellenzelle",
        "tableCellVerticalAlign": "Vertikale Ausrichtung der Tabellenzelle",
        "createTable": "Tabelle erstellen",
        "removeTable": "Tabelle entfernen",
        "tableHeader": "Tabellenkopfzeile",
        "tableRemove": "Tabelle entfernen",
        "tableCellBackground": "Tabellenzellenhintergrund",
        "tableEditProperties": "Eigenschaften der Tabellenbearbeitung",
        "styles": "Styles",
        "insertColumnLeft": "Spalte links einfügen",
        "insertColumnRight": "Spalte rechts einfügen",
        "deleteColumn": "Spalte löschen",
        "insertRowBefore": "Zeile vor einfügen",
        "insertRowAfter": "Zeile einfügen nach",
        "deleteRow": "Zeile löschen",
        "tableEditHeader": "Tabelle bearbeiten",
        "TableHeadingText": "Überschrift",
        "TableColText": "Col",
        "imageInsertLinkHeader": "Link einfügen",
        "editImageHeader": "Bild bearbeiten",
        "alignmentsDropDownLeft": "Linksbündig",
        "alignmentsDropDownCenter": "Im Zentrum anordnen",
        "alignmentsDropDownRight": "Rechts ausrichten",
        "alignmentsDropDownJustify": "Justize ausrichten",
        "imageDisplayDropDownInline": "In der Reihe",
        "imageDisplayDropDownBreak": "Brechen",
        "tableInsertRowDropDownBefore": "Reihe vorher einfügen",
        "tableInsertRowDropDownAfter": "Zeile danach einfügen",
        "tableInsertRowDropDownDelete": "Zeile löschen",
        "tableInsertColumnDropDownLeft": "Spalte links einfügen",
        "tableInsertColumnDropDownRight": "Spalte rechts einfügen",
        "tableInsertColumnDropDownDelete": "Spalte löschen",
        "tableVerticalAlignDropDownTop": "Top ausrichten",
        "tableVerticalAlignDropDownMiddle": "Mitte ausrichten",
        "tableVerticalAlignDropDownBottom": "Unten ausrichten",
        "tableStylesDropDownDashedBorder": "Gestrichelte Grenzen",
        "tableStylesDropDownAlternateRows": "Alternative Zeilen",
        "pasteFormat": "Format einfügen",
        "pasteFormatContent": "Wählen Sie die Formatierungsaktion aus",
        "plainText": "Einfacher Text",
        "cleanFormat": "sauber",
        "keepFormat": "Behalten",
        "formatsDropDownParagraph": "Absatz",
        "formatsDropDownCode": "Kodex",
        "formatsDropDownQuotation": "Zitat",
        "formatsDropDownHeading1": "Überschrift 1",
        "formatsDropDownHeading2": "Überschrift 2",
        "formatsDropDownHeading3": "Überschrift 3",
        "formatsDropDownHeading4": "Überschrift 4",
        "fontNameSegoeUI": "Segoe UI",
        "fontNameArial": "Arial",
        "fontNameGeorgia": "Georgia",
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        "fontNameImpact": "Einschlag'",
        "fontNameTahoma": "Tahoma",
        "fontNameTimesNewRoman": "Mal Neu römisch",
        "fontNameVerdana": "Verdana"
      }
    }
  });
  Vue.use(RichTextEditorPlugin);
  export default {
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs11" %}

RTL

Specifies the direction of the Rich Text Editor component using the enableRtl property. For writing systems that require it like Arabic, Hebrew, etc., the direction can be switched to right-to-left.

Note: It will not change based on the locale property.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340" enableRtl=true>
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
```

```
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs12" %}

Miscellaneous in Vue Rich text editor component

Placeholder

Specifies the placeholder for the Rich Text Editor's content used when the Rich Text Editor body is empty through the placeholder property.

Through the e-rte-placeholder class to define our custom font family, font color, and styles to the placeholder text.

Copyright © 2001 -2024 Syncfusion Inc.

` css

.e-richtexteditor .e-rte-placeholder {

font-family: monospace;

}

`

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="200" placeholder="Type
Something">
  </ejs-richtexteditor>
</template>
<style>
  .e-richtexteditor .rte-placeholder {
    font-family: monospace;
  }
</style>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs24" %}

Character count

The Rich Text Editor automatically counts the number of characters in the content are while typing using the showCharCount property. The characters count displayed at the bottom of the editor. You can limit the number of characters in your content using the maxLength property. By default, the editor sets the characters limit value is infinity.

Copyright © 2001 -2024 Syncfusion Inc.

The character count color will be modified based on the characters in the Rich Text Editor.

| **Status** | **Description** |

| --- | --- |

| Normal | Till 70% of given maxLength value, character count color is black. |

| Warning | Once the number of character count in the Rich Text Editor reached 70% of given maxLength value, the character count color will be orange, indicating that, the count value going to reach the maximum count. |

| Error | Once the number of character count in the Rich Text Editor reached 90% of given maxLength value, the character count color will be red, indicating that, the count value reached the maximum count. |

To create Rich Text Editor with character count feature, inject the Count module to the Rich Text Editor using the RichTextEditor.Inject(Count) method.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340"
:showCharCount="showCharCount"
  :maxLength="maxLength" :value="value"></ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
     data: function() {
      return {
        showCharCount: true,
        maxLength: 2000,
        value: `<p>The Rich Text Editor component is WYSIWYG ("what you see
is what you get") editor that provides the best user experience to create
and update the content. Users can format their content using standard
toolbar commands.</p>
          <p><b>Key features:</b></p>
          <ul>
            <li>
              <p>Provides IFRAME and DIV modes</p>
            </li>
            <li>
              <p>Capable of handling markdown editing.</p>
            </li>
            <li>
              <p>Contains a modular library to load the necessary
functionality on demand.</p>
            </li>
            <li>
              <p>Provides a fully customizable toolbar.</p>
            </li>
            <li>
              <p>Provides HTML view to edit the source directly for
developers.</p>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            </li>
            <li>
              <p>Supports third-party library integration.</p>
            </li>
            <li>
              <p>Allows preview of modified content before saving it.</p>
            </li>
            <li>
              <p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
            </li>
          </ul>`,
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs25" %}

Code view

Rich Text Editor includes the ability for users to directly edit HTML code via Source View in the text area. If you made any modification in Source view directly, the changes will be reflected in the Rich Text Editor's content. So, the users will have more flexibility over the content they have created.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="rteObj" :value="value" :height="340"
:toolbarSettings="toolbarSettings" :actionComplete="actionCompleteHandler"
:showCharCount="showCharCount" :maxLength="maxLength"></ejs-richtexteditor>
</template>
<script>
  import Vue from "vue";
  import { Browser, addClass, removeClass } from "@syncfusion/ej2-base";
  import { RichTextEditorPlugin, Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
  Vue.use(RichTextEditorPlugin);
  export default {
    data: function() {
```

```
      return {
        showCharCount: true,
        myCodeMirror: '',
        value: `<p>The Rich Text Editor component is WYSIWYG ("what you see
is what you get") editor that provides the best user experience to create
and update the content. Users can format their content using standard
toolbar commands.</p>
          <p><b>Key features:</b></p>
          <ul>
            <li>
              <p>Provides IFRAME and DIV modes</p>
            </li>
            <li>
              <p>Capable of handling markdown editing.</p>
            </li>
            <li>
              <p>Contains a modular library to load the necessary
functionality on demand.</p>
            </li>
            <li>
              <p>Provides a fully customizable toolbar.</p>
            </li>
            <li>
              <p>Provides HTML view to edit the source directly for
developers.</p>
            </li>
            <li>
              <p>Supports third-party library integration.</p>
            </li>
            <li>
              <p>Allows preview of modified content before saving it.</p>
            </li>
            <li>
              <p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
            </li>
          </ul>`,
        maxLength: 2000,
        toolbarSettings: {
          items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
            'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
            'LowerCase', 'UpperCase', '|',
            'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
            'Outdent', 'Indent', '|',
            'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
'SourceCode',
            'FullScreen', '|', 'Undo', 'Redo'
          ]
        },
      };
    },
    methods: {
      mirrorConversion: function(e) {
        var textArea =
this.$refs.rteObj.ej2Instances.contentModule.getEditPanel();
        var id = this.$refs.rteObj.ej2Instances.getID() +  'mirror-view';
```

```
        var mirrorView =
this.$refs.rteObj.ej2Instance.inputElement.parentElement.querySelector('#' +
id);
        var charCount
=this.$refs.rteObj.ej2Instance.element.querySelector('.e-rte-character-
count');
        if (e.targetItem === 'Preview') {
          textArea.style.display = 'block';
          mirrorView.style.display = 'none';
          textArea.innerHTML = this.myCodeMirror.getValue();
          charCount.style.display = 'block';
        } else {
          if (!mirrorView) {
            mirrorView = document.createElement('div', { className: 'e-
content' });
            mirrorView.id = id;
            textArea.parentNode.appendChild(mirrorView);
          } else {
            mirrorView.innerHTML = '';
          }
          textArea.style.display = 'none';
          mirrorView.style.display = 'block';
          this.renderCodeMirror(mirrorView,
this.$refs.rteObj.ej2Instances.value);
          charCount.style.display = 'none';
        }
      },
      renderCodeMirror: function(mirrorView, content) {
        this.myCodeMirror = CodeMirror(mirrorView, {
          value: content,
          lineNumbers: true,
          mode: 'text/html',
          lineWrapping: true,
        });
      },
      actionCompleteHandler: function(e) {
        if (e.targetItem && (e.targetItem === 'SourceCode' || e.targetItem
=== 'Preview')) {

this.$refs.rteObj.ej2Instances.sourceCodeModule.getPanel().style.display =
'none';
          this.mirrorConversion(e);
        }
        else {
          var proxy = this;
          setTimeout(function () {

proxy.$refs.rteObj.ej2Instances.toolbarModule.refreshToolbarOverflow();
          }, 400);
        }
      }
    },
    provide:{
      richtexteditor:[Toolbar, Link, Image, Count, HtmlEditor, QuickToolbar]
    }
  }
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
  .e-code-mirror::before {
    content: '\e345';
  }
  .e-html-preview::before {
    content: '\e350';
  }
  .CodeMirror-linenumber,
  .CodeMirror-gutters {
    display: none;
  }
</style>
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
    <title>EJ2 Vue Sample</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Typescript UI Controls" />
    <meta name="author" content="Syncfusion" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/material.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/codemirror.js"
type="text/javascript"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/css/css.js
" type="text/javascript"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/xml/xml.js
" type="text/javascript"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/htmlmixed/
htmlmixed.js" type="text/javascript"></script>
    <link href="index.css" rel="stylesheet" />
    <script src="systemjs.config.js"></script>
</head>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<body>
    <div id='app'>Loading....</div>
</body>
</html>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/code-mirror-cs1" %}

Undo/Redo manager

Undo and redo tools allow you to edit the text by disregard/cancel the recently made changes and restore it to previous state. It is a useful tool to restore the performed action which got changed by mistake. By default, upto 30 actions can be undo/redo in the editor.

To undo and redo operations, do one of the following:

- Press the undo/redo button on the toolbar.
- Press the **Ctrl + Z/Ctrl + Y** combination on the keyboard.

Customize the undo/redo step count using the undoRedoSteps property. By default, undo/redo actions take 300ms time interval for store the action to the undoRedoManager. The time interval can be customized by using the undoRedoTimer.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340"
:toolbarSettings="toolbarData" :UndoRedoSteps="50" :UndoRedoTimer="400">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
```

```
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        toolbarData: {
          items: ['Undo', 'Redo']
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs26" %}

Prevention of cross-site scripting (XSS)

The Rich Text Editor allows the users to edit the content with security by preventing cross-site scripting (XSS). By default, provided built-in support to remove the elements from editor content, which cause XSS attack. The editor removes the elements based on the attributes if it is possible to execute script.

In the following sample, removed `script` tag and `onmouseover` attribute from content of the Rich Text Editor.

APP.VUE

```
<template>
   <ejs-richtexteditor :value="value"></ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        value: `<div onmouseover='javascript:alert(1)'>Prevention of Cross
Sit Scripting (XSS) </div> <script>alert('hi')<\/script>`
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs27" %}

It is only applicable to editorMode as HTML.

Custom cross-site scripting

You can also filter the elements and attributes additionally, which cause the XSS attack through beforeSanitizeHtml event. Return the value from the event argument helper function to apply in the editor. To prevent the built-in support and make own cross-site scripting rules, set cancel argument to true.

The following sample demonstrates how to filter script tag from value.

APP.VUE

```
<template>
   <ejs-richtexteditor :value="value"
:beforeSanitizeHtml="beforeSanitizeHtml"></ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  import { detach } from "@syncfusion/ej2-base";
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
```

```
        value: `<div>Prevention of Cross Sit Scripting (XSS) </div>
<script>alert('hi')<\/script>`
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    },
    methods: {
    beforeSanitizeHtml: function(e) {
      e.helper = value => {
        e.cancel = true;
        let temp = document.createElement("div");
        temp.innerHTML = value;
        let scriptTag = temp.querySelector("script");
        if (scriptTag) {
          detach(scriptTag);
        }
        return temp.innerHTML;
      };
    }
  }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs28" %}

Resizable support

This feature allows the editor to be resized dynamically. The users can enable or disable this feature using the enableResize property in the Rich Text Editor. If enableResize is set to true, the Rich Text Editor component creates grip at the bottom right corner, which allows resizing the component in the diagonal direction. The following sample demonstrates the resizable feature.

Enabling the resizable support

To render the Rich Text Editor in the resizable mode, set the enableResize property to true. The above feature is built as module and hence the Resize module needs to be included in your application.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="sample-container">
            <div class="default-section">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <ejs-richtexteditor :enableResize="true"><p>The Rich Text Editor
component is WYSIWYG ("what you see is what you get") editor that provides
the best user experience to create and update the content.
                    Users can format their content using standard toolbar
commands.</p>
            </ejs-richtexteditor>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Image, Resize, HtmlEditor }
from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
    provide:{
        richtexteditor:[Toolbar, Image, Resize HtmlEditor, Link]
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs12" %}

Specifying the Minimum and Maximum width and height for Resize

To have a restricted resizable area for the Rich Text Editor, you need to specify the min-width, max-width, min-height, and max-height CSS properties for the control's wrapper element. By default, the control is capable of resizing upto the current viewport. The e-richtexteditor CSS class will be available in the component's wrapper and can be used for applying the above mentioned styles.

APP.VUE

```
<template>
    <div class="control-section">
        <div class="sample-container">
            <div class="default-section">
            <ejs-richtexteditor :enableResize="true"><p>The Rich Text Editor
component is WYSIWYG ("what you see is what you get") editor that provides
the best user experience to create and update the content.
                    Users can format their content using standard toolbar
commands.</p>
            </ejs-richtexteditor>
            </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </div>
    </div>
</template>
<style>
  .e-richtexteditor {
      min-width: 200px;
      max-width: 800px;
      min-height: 100px;
      max-height: 300px;
  }
</style>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Image, Resize, HtmlEditor }
from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
    provide:{
        richtexteditor:[Toolbar, Image, Resize HtmlEditor, Link]
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs13" %}

Number and Bullet Format Lists

This feature allows the user to change the appearance of the Numbered and Bulleted lists. Users can also apply different numbering or bullet formats lists such as lowercase greek, upper Alpha, square and circles. You can also customize the style type of the lists to be populated in the dropdown from the toolbar by using the `numberFormatList` and `bulletFormatList` properties in the Rich Text Editor.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340"
:toolbarSettings="toolbarData">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        toolbarData: {
          items: ['NumberFormatList', 'BulletFormatList']
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs29" %}

Xhtml validation in Vue Rich text editor component

The editor provides an option to validate the source content of the Rich Text Editor against the XHTML standard using the 'enableXhtml' property. When you enter or modify content in the editor, it continuously checks the XHTML source content and removes elements and attributes that are not valid.

The editor checks the following settings on validation:

Attributes

- Must be specified in lowercase.
- Proper use of quotation marks around the attributes.
- Must be valid attributes for corresponding HTML element.
- All the required attributes must be included in the HTML element.

HTML Elements

- Must be in lowercase.
- All opening tags must be closed.
- Allows only the valid HTML elements.
- Elements must be properly nested.
- All elements must have one root element.
- Should not use inline elements inside the block elements.

APP.VUE

```
<template>
    <ejs-richtexteditor ref="defaultRTE" :height="400" enableXhtml=true>
        <p>The Rich Text Editor component is WYSIWYG ("what you see is what
you get") editor that provides the best user experience to create and update
the content.Users can format their content using standard toolbar
commands.</p>
        <p><b>Key features:</b></p>
        <ul>
            <li><p>Provides IFRAME and DIV modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            <li><p>Contains undo/redo manager.</p></li>
            <li><p>Creates bulleted and numbered lists.</p></li>
        </ul>
    </ejs-richtexteditor>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
    import Vue from "vue";
    import { RichTextEditorPlugin, Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
    Vue.use(RichTextEditorPlugin);
    export default {
        provide: {
            richtexteditor:[Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar]
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs14" %}

Inline mode in Vue Rich text editor component

This is the inline example for the Rich Text Editor. For this you must set the inlineMode property.

Inline edition allows you to select any editable element or click the element on the page and edit it in-place.

Inline editing is a true WYSIWYG formation and on the contrary to Rich Text Editor HTML/MD editing, the styles that are used for edited content comes directly from the document stylesheet. This means that inline editors ignore the default Rich Text Editor content styles.

Show on select/click

Enabling the onSelection option of inlineMode makes the inline Rich Text Editor to appear. You can select the text in the editable area otherwise the inline Rich Text Editor will be appear once click into the editable area.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340"
:toolbarSettings="toolbarData" :inlineMode="inlineModeData">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        toolbarData: {
          items: ['Bold', 'Italic', 'Underline', 'Formats', '-',
'Alignments', 'OrderedList', 'UnorderedList', 'CreateLink']
        },
        inlineModeData: {
          enable: true,
          onSelection: true
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs20" %}

See Also

- How to edit the quick toolbar settings
- How to insert link editing option in the toolbar items
- How to insert image editing option in the toolbar items

Paste cleanup in Vue Rich text editor component

The Rich Text Editor allows you to reduce the effort while converting the Microsoft Word content to HTML format with format and styles.

MS Word to HTML

By default, Rich Text Editor consider the following processes on paste content from Microsoft Word.

List conversion: The list elements copied from the Microsoft Word document contains paragraph tags with styles and classes. The list elements are converted to standard HTML list elements by referring the styles and class names in the paragraph tags.

Converting style: The styles of the elements copied from the Microsoft Word document are converted to standard CSS styles and added as inline styles for each respective element.

Tags and comments: The Microsoft Word specific XML tags and comments are removed when cleanup on paste.

Paste cleanup

You can control the formatting and styles on pasting the content to the editor using the pasteCleanup settings property. The following settings are available to clean up the content:

| API | Description | Default Value | Type |

|:---------------:|:---------:|:--------------------------:|:---------:|

| prompt | To invoke prompt dialog with paste options on pasting the content in editor. | false | boolean |

| plainText | To paste the content as plain text. | false | boolean |

| keepFormat | To keep the same format with copied content. | true | boolean |

| deniedTags | To ignore the tags when pasting HTML content. | null | string[] |

| deniedAttrs | To paste the content by filtering out these attributes from the content. | null | string[] |

| allowedStyleProps | To paste the content by accepting these style attributes and removing other style attributes. | ['background', 'background-color', 'border', 'border-bottom', 'border-left', 'border-radius', 'border-right', 'border-style', 'border-top', 'border-width', 'clear', 'color', 'cursor', 'direction', 'display', 'float', 'font', 'font-family', 'font-size', 'font-weight', 'font-style', 'height', 'left', 'line-height', 'margin', 'margin-top', 'margin-left', 'margin-right', 'margin-bottom', 'max-height', 'max-width', 'min-height', 'min-

Copyright © 2001 -2024 Syncfusion Inc.

width', 'overflow', 'overflow-x', 'overflow-y', 'padding', 'padding-bottom', 'padding-left', 'padding-right', 'padding-top', 'position', 'right', 'table-layout', 'text-align', 'text-decoration', 'text-indent', 'top', 'vertical-align', 'visibility', 'white-space', 'width'] | string[] |

Rich Text Editor features are segregated into individual feature-wise modules. To use paste cleanup, inject paste cleanup module using the RichTextEditor.Inject(PasteCleanup).

Prompt dialog

When prompt is set to true, pasting the content in the editor will open a dialog box that contains three options Keep, Clean, and Plain Text as radio buttons:

1. Keep: Radio button to keep the same format with copied content.
2. Clean: Radio button to clear all the style formats with copied content.
3. Plain Text: Radio button to paste the copied content as plain text without any formatting or style (including the removal of all tags).

When prompt value is set true, the API properties plainText and keepFormat will not be considered for processing when pasting the content.

Paste as plain text

When plainText is set to true, the copied content will be converted as plain text by removing all the HTML tags and styles applied to it and only the plain text is pasted in the editor.

When plainText value is set true, the API property prompt should be set to false, and keepFormat will not be considered for processing when pasting the content.

Keep format

When keepFormat is set to true, the copied content will maintain all the style formatting allowed in the allowedStyleProps on pasting the content in the editor.

When keepFormat is set to false, the style in the copied content will be removed without considering the allowed styles in the allowedStyleProps when pasting the content in the editor.

When keepFormat value is set true, the API property prompt and plainText should be set to false.

Denied tags

When deniedTags values are set, the tags that matches the 'denied tags' list will be removed on pasting the copied content in the editor. For Example,

1. ‘a’: Paste the content by filtering out anchor tags.
2. ‘a[!href]’: Paste the content by filtering out anchor tags that do not have the ‘href’ attribute.
3. ‘a[href, target]’: Paste the content by filtering out anchor tags that have the ‘href’ and ‘target’ attributes.

Denied attributes

When the deniedAttrs values are set, the attributes that matches the 'denied attributes' list will be removed on pasting the copied content in the editor. For Example,

‘id’, ‘title’: This will remove the attributes ‘id’ and ‘title’ from all tags.

Copyright © 2001 -2024 Syncfusion Inc.

Allowed style properties

By default, the following basic styles are allowed on pasting the content to the editor.

['background', 'background-color', 'border', 'border-bottom', 'border-left', 'border-radius', 'border-right', 'border-style', 'border-top', 'border-width', 'clear', 'color', 'cursor', 'direction', 'display', 'float', 'font', 'font-family', 'font-size', 'font-weight', 'font-style', 'height', 'left', 'line-height', 'margin', 'margin-top', 'margin-left', 'margin-right', 'margin-bottom', 'max-height', 'max-width', 'min-height', 'min-width', 'overflow', 'overflow-x', 'overflow-y', 'padding', 'padding-bottom', 'padding-left', 'padding-right', 'padding-top', 'position', 'right', 'table-layout', 'text-align', 'text-decoration', 'text-indent', 'top', 'vertical-align', 'visibility', 'white-space', 'width']

When you configure allowedStyleProps, the styles, which matches the 'allowed style properties' list are allowed, all other style properties will be removed on pasting the content in the editor.

For Example,

allowedStyleProps: ['color', 'margin']': This will allow only the style properties 'color' and 'margin' in each pasted element.

In the following example, the paste cleanup related settings are explained with its module configuration

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj" :toolbarSettings="toolbarSettings"
:pasteCleanupSettings="pasteCleanupSettings"><p>Rich Text Editor is a
WYSIWYG editing control which will reduce the effort for users while trying
to express their formatting word content as HTML or Markdown format.</p>
        <p><b>Paste Cleanup properties:</b></p>
        <ul>
            <li>
                <p>prompt - specifies whether to enable the prompt when
pasting in Rich Text Editor.</p>
            </li>
            <li>
                <p>plainText - specifies whether to paste as plain text or
not in Rich Text Editor.</p>
            </li>
            <li>
                <p>keepFormat- specifies whether to keep or remove the
format when pasting in Rich Text Editor.</p>
            </li>
            <li>
                <p>deniedTags - specifies the tags to restrict when pasting
in Rich Text Editor.</p>
            </li>
            <li>
                <p>deniedAttributes - specifies the attributes to restrict
when pasting in Rich Text Editor.</p>
            </li>
            <li>
                <p>allowedStyleProperties - specifies the allowed style
properties when pasting in Rich Text Editor.</p>
```

```
            </li>
        </ul></ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, HtmlEditor, PasteCleanup } from
"@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
            toolbarSettings: {
                type: 'Expand',
                items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
                'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
                'LowerCase', 'UpperCase', '|',
                'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
                'Outdent', 'Indent', '|',
                'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
                'SourceCode', 'FullScreen', '|', 'Undo', 'Redo'
            ]
            },
            pasteCleanupSettings: {
                prompt: true,
                plainText: false,
                keepFormat: false,
                deniedTags: ['a'],
                deniedAttrs: ['class', 'title', 'id'],
                allowedStyleProps: ['color', 'margin', 'font-size']
            }
        };
    },
    provide:{
        richtexteditor:[Toolbar, HtmlEditor, PasteCleanup]
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs14" %}

Copyright © 2001 -2024 Syncfusion Inc.

Mention integration in Vue Rich text editor component

By integrating the Mention component with a Rich Text Editor, users can easily mention or tag other users or objects from the suggested list without having to manually type out their names or other identifying information.

The target property of the Mention component allows you to specify the ID of the content editable div element within the Rich Text Editor that you want to bind the Mention component to. This allows you to enable the Mention functionality within the Rich Text Editor, so that users can mention or tag other users or objects from the suggested list while editing the text.

When the user types the @ symbol followed by a character, the Rich Text Editor will display a list of suggestions for items that the user can select from. The user can then select an item from the list by clicking on it, or by typing the name of the item they want to tag.

In the following sample, configured the following properties with popup dimensions.

- allowSpaces - Allow to continue search action if user enter space after mention character while searching.
- suggestionCount - The maximum number of items that will be displayed in the suggestion list.
- itemTemplate - Used to display the customized appearance in suggestion list.

APP.VUE

```
<template>
    <div>
        <ejs-richtexteditor id="mention_integration"  placeholder="Type @
and tag the name" :actionBegin="onActionBegin">
            <p>Hello <span contenteditable="false" class="e-mention-chip"><a
href="mailto:maria@gmail.com" title="maria@gmail.com">@Maria</a></span></p>
            <p>Welcome to the mention integration with rich text editor
demo. Type <code>@</code> character and tag user from the suggestion list.
</p>
        </ejs-richtexteditor>
        <ejs-mention id="mentionEditor" target="#mention_integration_rte-
edit-view" :suggestionCount="8" :showMentionChar="false" :allowSpaces="true"
:dataSource="data" :fields="fieldsData" popupWidth='250px'
popupHeight='200px' :itemTemplate="'iTemplate'"
:displayTemplate="'dTemplate'">
            <template v-slot:iTemplate="{data}">
                <table><tr><td><div id="mention-TemplateList"><img
class="mentionEmpImage" :src="data.EmployeeImage" alt="employee" /><span
:class="'e-badge e-badge-success e-badge-overlap e-badge-dot e-badge-
bottom'+ data.Status"></span></div></td><td class="mentionNameList"><span
class="person">{{data.Name}}</span><span
class="email">{{data.EmailId}}</span></td></tr></table>
            </template>
            <template v-slot:dTemplate="{data}">
                <a :href="'mailto:' + data.EmailId"
:title="data.EmailId">@{{data.Name}}</a>
            </template>
        </ejs-mention>
    </div>
</template>
<script>
```

```
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Image, QuickToolbar,
HtmlEditor } from "@syncfusion/ej2-vue-richtexteditor";
import { MentionPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(MentionPlugin);
Vue.use(RichTextEditorPlugin);
export default {
    data: function() {
        return {
          data: [
              { Name: "Selma Rose", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/2.png",
EmailId: "selma@gmail.com" },
              { Name: "Maria", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/1.png",
EmailId: "maria@gmail.com" },
              { Name: "Russo Kay", Status: "busy", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/8.png",
EmailId: "russo@gmail.com" },
              { Name: "Camden Kate", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/9.png",
EmailId: "camden@gmail.com" },
              { Name: "Robert", Status: "busy", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/dp.png",
EmailId: "robert@gmail.com" },
              { Name: "Garth", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/7.png",
EmailId: "garth@gmail.com" },
              { Name: "Andrew James", Status: "away", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/pic04.png",
EmailId: "andrew@gmail.com" },
              { Name: "Olivia", Status: "busy", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/5.png",
EmailId: "olivia@gmail.com" },
              { Name: "Sophia", Status: "away", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/6.png",
EmailId: "sophia@gmail.com" },
              { Name: "Margaret", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/3.png",
EmailId: "margaret@gmail.com" },
              { Name: "Ursula Ann", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/dp.png",
EmailId: "ursula@gmail.com" },
              { Name: "Laura Grace", Status: "away", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/4.png",
EmailId: "laura@gmail.com" },
              { Name: "Albert", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/pic03.png",
EmailId: "albert@gmail.com" },
              { Name: "William", Status: "away", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/8.png",
EmailId: "william@gmail.com" }
            ],
            fieldsData: { text: 'Name' }
        };
    },
    provide:{
```

```
        richtexteditor:[Toolbar, Link, Image, QuickToolbar, HtmlEditor]
    },
    methods: {
       onActionBegin: function(args) {
            if (args.requestType === 'EnterAction') {
                args.cancel = true;
            }
        }
    }
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/bootstrap5.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
#mention-TemplateList {
        position: relative;
        display: inline-block;
        padding: 2px;
    }
    .mentionNameList .person, .mentionNameList.email {
        display: block;
        line-height: 20px;
        text-indent: 5px;
    }
    .mentionNameList .person {
        font-size: 16px;
    }
    .mentionEmpImage {
        display: inline-block;
        width: 46px;
        height: 46px;
        padding: 3px;
        border-radius: 25px;
    }
    #mention-TemplateList .e-badge-success {
        left: 76%;
        bottom: 4px;
        top: auto;
    }
    #mention_integration_rte-edit-view_popup .e-dropdownbase .e-list-item {
        line-height: 8px;
    }
    #mention-TemplateList .e-badge-success {
        background-color: #4d841d;
        color: #fff;
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    #mention-TemplateList .e-badge-success.away {
        background-color: #fedd2d;
        color: #fff;
    }
    #mention-TemplateList .e-badge-success.busy {
        background-color: #de1a1a;
        color: #fff;
    }
    #mention-TemplateList .e-badge.e-badge-dot {
        height: 10px;
        width: 10px;
    }
    #mention_integration .e-mention-chip{
        cursor: pointer;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/mention-integration-cs1" %}

View Sample

See Also

- Mention

Enter key in Vue Rich text editor component

Rich Text Editor allows to customize the tag that is inserted when pressing the enter key and shift + enter key in the Rich Text Editor.

Enter key customization

By default, the <p> tag will be created while pressing the enter key. The enter key can be customized by using the enterKey property, where the possible tags that can be used to customize are <p>, , and <br>.

When the enter key is customized with any of the possible values, pressing the enter key in the editor will create a new tag that is configured. Also, when the enter key is configured the default value of the Rich Text Editor will also change respectively with the configured values.

APP.VUE

```
<template>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
            <table class="api">
                <tbody>
                    <tr>
                        <td>
                            <div>
                                <ejs-dropdownlist ref="enterOptionInstance"
:dataSource="enterData" :fields='fields' :popupHeight="height"
:change="enterChange" :value='enterValue' :placeholder='enterPlaceHolder'
:floatLabelType='floatLabelType'></ejs-dropdownlist>
                            </div>
                        </td>
```

```
                    </tr>
                </tbody>
            </table>
            <br/>
            <ejs-richtexteditor ref="rteInstance" :height="height">
                <p>In Rich text Editor, the enter key and shift + enter key
actions can be customized using the enterKey and shiftEnterKey APIs. And the
possible values are as follows:</p><ul><li>P - When 'P' is configured,
pressing enter or shift + enter will create a 'p' tag</li><li>DIV - When
'DIV' is configured, pressing enter or shift + enter will create a 'div'
tag</li><li>BR - When 'BR' is configured, pressing enter or shift + enter
will create a 'br' tag</li></ul>
            </ejs-richtexteditor>
        </div>
    </div>
</div>
</template>
<style>
    .api {
        width: 50%;
    }
    .api td {
        padding-right: 100px;
        margin-top: 3px;
    }
    @media only screen and (max-width: 600px) {
        .api {
            width: 100%;
        }
        .api td {
            display: block;
        }
    }
    @media only screen and (min-width: 600px) and (max-width: 1000px) {
        .api {
            width: 50%;
        }
    }
</style>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";

  Vue.use(DropDownListPlugin);
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        height: 250,
        enterPlaceHolder: 'When pressing the enter key',
        floatLabelType: 'Always',
        enterData:  [
            { text: "Create a new <p>", value: "P" },
            { text: "Create a new <div>", value: "DIV" },
            { text: "Create a new <br>", value: "BR" }
```

```
        ],
        fields: { text: 'text', value: 'value' },
        enterValue: 'P',
      };
    },
    methods: {
        enterChange: function() {
            if (this.$refs.enterOptionInstance.ej2Instances.value === 'P') {
                this.$refs.rteInstance.ej2Instances.enterKey = 'P';
                 this.$refs.rteInstance.ej2Instances.value = `<p>In Rich
text Editor, the enter key and shift + enter key actions can be customized
using the enterKey and shiftEnterKey APIs. And the possible values are as
follows:</p><ul><li>P - When 'P' is configured, pressing enter or shift +
enter will create a 'p' tag</li><li>DIV - When 'DIV' is configured, pressing
enter or shift + enter will create a 'div' tag</li><li>BR - When 'BR' is
configured, pressing enter or shift + enter will create a 'br'
tag</li></ul>`;
            } else if (this.$refs.enterOptionInstance.ej2Instances.value ===
'DIV') {
                this.$refs.rteInstance.ej2Instances.enterKey = 'DIV';
                this.$refs.rteInstance.ej2Instances.value = `<div>In Rich
text Editor, the enter key and shift + enter key actions can be customized
using the enterKey and shiftEnterKey APIs. And the possible values are as
follows:</div><ul><li>P - When 'P' is configured, pressing enter or shift +
enter will create a 'p' tag</li><li>DIV - When 'DIV' is configured, pressing
enter or shift + enter will create a 'div' tag</li><li>BR - When 'BR' is
configured, pressing enter or shift + enter will create a 'br'
tag</li></ul>`;
            } else if (this.$refs.enterOptionInstance.ej2Instances.value ===
'BR') {
                this.$refs.rteInstance.ej2Instances.enterKey = 'BR';
                this.$refs.rteInstance.ej2Instances.value = `In Rich text
Editor, the enter key and shift + enter key actions can be customized using
the enterKey and shiftEnterKey APIs. And the possible values are as
follows:<ul><li>P - When 'P' is configured, pressing enter or shift + enter
will create a 'p' tag</li><li>DIV - When 'DIV' is configured, pressing enter
or shift + enter will create a 'div' tag</li><li>BR - When 'BR' is
configured, pressing enter or shift + enter will create a 'br'
tag</li></ul>`;
            }
        }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs3" %}

Shift-Enter key customization

By default, the <br> tag will be created while pressing the shift + enter key. The shift + enter key can be customized by using the shiftEnterKey property where the possible tags that can be used to customize are <br>, <p>, .

When the shift + enter key is customized with any of the possible values, pressing the shift + enter key in the editor will create a new tag that is configured. Also, when the shift + enter key is configured the default value of the Rich Text Editor will change respectively with the configured values.

APP.VUE

```
<template>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
            <table class="api">
                <tbody>
                    <tr>
                        <td>
                            <div>
                                <ejs-dropdownlist
ref="shiftEnterOptionInstance" :dataSource="shiftEnterData" :fields='fields'
:popupHeight="height" :change="shiftEnterChange" :value='shiftEnterValue'
:placeholder='shiftEnterPlaceHolder' :floatLabelType='floatLabelType'></ejs-
dropdownlist>
                            </div>
                        </td>
                    </tr>
                </tbody>
            </table>
            <br/>
            <ejs-richtexteditor ref="rteInstance" :height="height">
                <p>In Rich text Editor, the enter key and shift + enter key
actions can be customized using the enterKey and shiftEnterKey APIs. And the
possible values are as follows:</p><ul><li>P - When 'P' is configured,
pressing enter or shift + enter will create a 'p' tag</li><li>DIV - When
'DIV' is configured, pressing enter or shift + enter will create a 'div'
tag</li><li>BR - When 'BR' is configured, pressing enter or shift + enter
will create a 'br' tag</li></ul>
            </ejs-richtexteditor>
        </div>
    </div>
</div>
</template>
<style>
    .api {
        width: 55%;
```

```
    }
    .api td {
        padding-right: 100px;
        margin-top: 3px;
    }
    @media only screen and (max-width: 600px) {
        .api {
            width: 100%;
        }
        .api td {
            display: block;
        }
    }
    @media only screen and (min-width: 600px) and (max-width: 1000px) {
        .api {
            width: 55%;
        }
    }
</style>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";

  Vue.use(DropDownListPlugin);
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        height: 220,
        shiftEnterPlaceHolder: 'When pressing the shift + enter key',
        floatLabelType: 'Always',
        shiftEnterValue: 'BR',
        shiftEnterData:  [
            { text: "Create a new <br>", value: "BR" },
            { text: "Create a new <div>", value: "DIV" },
            { text: "Create a new <p>", value: "P" }
        ],
        fields: { text: 'text', value: 'value' }
      };
    },
    methods: {
        shiftEnterChange: function() {
            if (this.$refs.enterOptionInstance.ej2Instances.value === 'BR')
{
                this.$refs.rteInstance.ej2Instances.shiftEnterKey = 'BR';
            } else if (this.$refs.enterOptionInstance.ej2Instances.value ===
'DIV') {
                this.$refs.rteInstance.ej2Instances.shiftEnterKey = 'DIV';
            } else if (this.$refs.enterOptionInstance.ej2Instances.value ===
'P') {
                this.$refs.rteInstance.ej2Instances.shiftEnterKey = 'P';
            }
        }
    },
    provide: {
```

```
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs4" %}

Iframe in Vue Rich text editor component

When the iframeSettings option is enabled, the Rich Text Editor creates the iframe element as the content area on control initialization; it is used to display and editing the content. In content area, the editor displays only the body tag of a <iframe> document.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="350"
:iframeSettings="iframeData">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        iframeData: {
          enable: true
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs16" %}

IFrame attributes

The editor allows you to pass an additional attribute to body tag of a <iframe> element using attributes fields of the iframeSettings property. This property contains name/value pairs in string format. It is used to override the default appearance of the content area.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="350"
:iframeSettings="iframeData">
```

```
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        iframeData: {
          enable: true,
          attributes: {
            readonly: 'readonly'
          }
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs17" %}

Adding external CSS/Script file

The editor offers you to add external CSS file to style the <iframe> element. Easily change the appearance of editor's content using an external CSS file using styles field in the iframeSettings property.

Likewise, add the external script file to the <iframe> element using the scripts field of iframeSettings to provide the additional functionalities to the Rich Text Editor.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="350"
:iframeSettings="iframeData">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b>  <-- Custom style applied here</p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        iframeData: {
          enable: true,
          resources: {
            scripts: ['myScript.js'], // Scripts
            styles: ['myStyle.css'] // Styles
          }
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/iframe-resources-cs1" %}

See Also

- How to change the editor mode

Third party integration in Vue Rich text editor component

The Rich Text Editor can be integrated with third-party to suite the application scenario.

Copyright © 2001 -2024 Syncfusion Inc.

CodeMirror Integration

Rich Text Editor comes with a basic HTML source editor through the view-source property. CodeMirror plugin can be used to highlight the syntax of HTML. CodeMirror plugin for Rich Text Editor makes editing of HTML source code with a pleasant experience.

Import necessary CSS and JS files of CodeMirror to the HTML page.

Required JS files of code mirror.

`ts

import CodeMirror from "https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/codemirror.js";

import CodeMirror from "https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/css/css.js";

import CodeMirror from "https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/xml/xml.js";

import CodeMirror from "https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/htmlmixed/htmlmixed.js";

import CodeMirror from "https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/javascript/javascript.js";

`

Required CSS file of code mirror

`ts

@import "https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/codemirror.min.css";

`

Add a custom icon for HTML source editor in the toolbar of Rich Text Editor using the template option of ToolbarSettings, define the code mirror plugins, and then pass the Rich Text Editor content as argument in the actionComplete event.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="rteObj" :height="360" :value="value"
:toolbarSettings="toolbarSettings" :actionComplete="actionCompleteHandler"
:showCharCount="showCharCount" :maxLength="maxLength"></ejs-richtexteditor>
</template>
<script>
  import Vue from "vue";
  import CodeMirror from
"https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/codemirror.js";
  import CodeMirror from
"https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/css/css.js";
  import CodeMirror from
"https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/xml/xml.js";
  import CodeMirror from
"https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/htmlmixed/html
mixed.js";
  import CodeMirror from
"https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/javascript/jav
ascript.js";
  import { Browser, addClass, removeClass } from "@syncfusion/ej2-base";
```

```
  import { RichTextEditorPlugin, Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
  Vue.use(RichTextEditorPlugin);
  export default {
    data: function() {
      return {
        showCharCount: true,
        myCodeMirror: '',
        value: `<p>The Rich Text Editor component is WYSIWYG ("what you see
is what you get") editor that provides the best user experience to create
and update the content. Users can format their content using standard
toolbar commands.</p>
          <p><b>Key features:</b></p>
          <ul>
            <li>
              <p>Provides IFRAME and DIV modes</p>
            </li>
            <li>
              <p>Capable of handling markdown editing.</p>
            </li>
            <li>
              <p>Contains a modular library to load the necessary
functionality on demand.</p>
            </li>
            <li>
              <p>Provides a fully customizable toolbar.</p>
            </li>
            <li>
              <p>Provides HTML view to edit the source directly for
developers.</p>
            </li>
            <li>
              <p>Supports third-party library integration.</p>
            </li>
            <li>
              <p>Allows preview of modified content before saving it.</p>
            </li>
            <li>
              <p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
            </li>
          </ul>`,
        maxLength: 2000,
        toolbarSettings: {
          items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
            'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
            'LowerCase', 'UpperCase', '|',
            'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
            'Outdent', 'Indent', '|',
            'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
'SourceCode',
            'FullScreen', '|', 'Undo', 'Redo'
          ]
        },
      };
    },
    methods: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      mirrorConversion: function(e) {
        var textArea =
this.$refs.rteObj.ej2Instances.contentModule.getEditPanel();
        var id = this.$refs.rteObj.getID() +  'mirror-view';
        var mirrorView = this.$refs.rteObj.$el.parentNode.querySelector('#'
+ id);
        var charCount =
this.$refs.rteObj.$el.parentNode.querySelectorAll('.e-rte-character-count');
        if (e.targetItem === 'Preview') {
          textArea.style.display = 'block';
          mirrorView.style.display = 'none';
          textArea.innerHTML = this.myCodeMirror.getValue();
          [charCount[0].style.display = 'block';
        } else {
          if (!mirrorView) {
            mirrorView = document.createElement('div', { className: 'e-
content' });
            mirrorView.id = id;
            textArea.parentNode.appendChild(mirrorView);
          } else {
            mirrorView.innerHTML = '';
          }
          textArea.style.display = 'none';
          mirrorView.style.display = 'block';
          this.renderCodeMirror(mirrorView,
this.$refs.rteObj.ej2Instances.value);
          charCount[0].style.display = 'none';
        }
      },
      renderCodeMirror: function(mirrorView, content) {
        this.myCodeMirror = CodeMirror(mirrorView, {
          value: content,
          lineNumbers: true,
          mode: 'text/html',
          lineWrapping: true,
        });
      },
      actionCompleteHandler: function(e) {
        if (e.targetItem && (e.targetItem === 'SourceCode' || e.targetItem
=== 'Preview')) {

this.$refs.rteObj.ej2Instances.sourceCodeModule.getPanel().style.display =
'none';
          this.mirrorConversion(e);
        }
        else {
          var proxy = this;
          setTimeout(function () {

proxy.$refs.rteObj.ej2Instances.toolbarModule.refreshToolbarOverflow();
          }, 400);
        }
      }
    },
    provide:{
      richtexteditor:[Toolbar, Link, Image, Count, HtmlEditor, QuickToolbar]
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
@import
"https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/codemirror.min.css"
;
  .e-code-mirror::before {
    content: '\e345';
  }
  .e-html-preview::before {
    content: '\e350';
  }
  .CodeMirror-linenumber,
  .CodeMirror-gutters {
    display: none;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/markdown-cs11" %}

Exec command in Vue Rich text editor component

In Rich Text Editor, execCommand used to perform commands for the modification of content in editable area.

The execCommand will perform the following commands.

| Commands | Description | Code snippets |

|---|---|---|

| bold | Bold the selected content in the Rich Text Editor. |rteObj.executeCommand('bold');|

| italic | The selected text will be italics. |rteObj.executeCommand('italic');|

| underline | Underline the selected text in the Rich Text Editor. |rteObj.executeCommand('underline');|

| strikeThrough | Apply single line strike through formatting for the selected text. |rteObj.executeCommand('strikeThrough');|

| superscript | Makes the selected text as superscript (higher). |rteObj.executeCommand('superscript');|

| subscript | Makes the selected text as subscript (lower). |rteObj.executeCommand('subscript');|

Copyright © 2001 -2024 Syncfusion Inc.

| uppercase | Change the case of selected text to upper in the content. |rteObj.executeCommand('uppercase');|

| lowercase | Change the case of selected text to lower in the content. |rteObj.executeCommand('uppercase');|

| fontColor | Apply the specified font color for the selected text. |rteObj.executeCommand('fontColor', 'yellow');|

| fontName | Apply the specified font name for the selected text. |rteObj.executeCommand('fontName', 'Arial');|

| fontSize | Apply the specified font size for the selected text. |rteObj.executeCommand('fontSize', '10pt');|

| formatBlock | Apply the specified format styles for the selected text. |rteObj.executeCommand('formatBlock', 'H1');|

| backColor | Apply the specified background color the selected text. | rteObj.executeCommand('backColor', 'red');|

| justifyCenter | Align the content with center margin. | rteObj.executeCommand('justifyCenter');|

| justifyFull | Align the content with justify margin. |rteObj.executeCommand('justifyFull');|

| justifyLeft | Align the content with left margin. | rteObj.executeCommand('justifyLeft');|

| justifyRight | Align the content with right margin. | rteObj.executeCommand('justifyLeft');|

| undo | Allows to undo the actions. | rteObj.executeCommand('undo');|

| createLink | Creates a hyperlink to a text or image to a specific location in the content. | rteObj.executeCommand('createLink',{ text: 'Links', url: 'http://', title : 'Link' }); |

| indent | Allows to increase the indent level of the content. | rteObj.executeCommand('indent');|

| insertHTML | Insert the html content to the current cursor position. | rteObj.executeCommand('insertHTML', 'inserted an html');|

| insertOrderedList | Create a new list item(numbered). | rteObj.executeCommand('insertOrderedList');|

| insertUnorderedList | Create a new list item(bulleted). |rteObj.executeCommand('insertUnorderedList');|

| outdent | Allows to decrease the indent level of the content. | rteObj.executeCommand('outdent');|

| redo | Allows to redo the actions | rteObj.executeCommand('redo');|

| removeFormat | remove all formatting styles (such as bold, italic, underline, color, superscript, subscript, and more) from currently selected text. |rteObj.executeCommand('removeFormat');|

| insertText | Insert text to the current cursor position. | rteObj.executeCommand('insertText', 'inserted a text'); |

Copyright © 2001 -2024 Syncfusion Inc.

| insertImage | Insert an image to the current cursor position. | rteObj.executeCommand('insertImage', { url: 'https://ej2.syncfusion.com/javascript/demos/src/rich-text-editor/images/RTEImage-Feather.png', cssClass: 'rte-img'}); |

| copyFormatPainter | Copy the format of selected text and apply it to another text. | rteObj.executeCommand('copyFormatPainter', formatPainterSettings);|

| applyFormatPainter | Apply the copied format to the selected text. | rteObj.executeCommand('applyFormatPainter');|

| escapeFormatPainter | Remove the previously copied format and disable the sticky mode | rteObj.executeCommand('escapeFormatPainter');|

Note: The 'ExecuteCommand' public method is not supported in Syncfusion Markdown Editor

Style in Vue Rich text editor component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the Rich Text Editor's content

Use the following CSS to customize the default Rich Text Editor's content properties like font-family, font-size and color.

`css

/ *To change font family and font size* /

.e-richtexteditor .e-rte-content .e-content,

.e-richtexteditor .e-source-content .e-content {

font-size: 20px;

font-family: Segoe ui;

}

/ *To change font color and content background* /

.e-richtexteditor .e-rte-content,

.e-richtexteditor .e-source-content {

background: seashell;

color: blue;

}

`

Customizing the Rich Text Editor's toolbar

Use the following CSS to customize the default color in the Rich Text Editor's toolbar icon.

`css

/ *To change font color for toolbar icon* /

.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-icons,

Copyright © 2001 -2024 Syncfusion Inc.

```
.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-icons:active {
color: red;
}
/ To change font color for toolbar button /
.e-toolbar .e-tbar-btn,
.e-toolbar .e-tbar-btn:active,
.e-toolbar .e-tbar-btn:hover {
color: red;
}
/ To change font color for toolbar button in active state/
.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-dropdown-btn.e-active .e-icons, .e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-dropdown-btn.e-active .e-rte-dropdown-btn-text {
color: red;
}
/ To change font color for expanded toolbar items /
.e-richtexteditor .e-rte-toolbar .e-toolbar-extended .e-toolbar-item .e-tbar-btn .e-icons,
.e-toolbar.e-extended-toolbar .e-toolbar-extended .e-toolbar-item .e-tbar-btn {
color: red;
}
```

Customizing the Rich Text Editor's character count

Use the following CSS to customize the default color in the Rich Text Editor's character count.

```
`css
/ To change font color, font family, font size and opacity  /
.e-richtexteditor .e-rte-character-count {
color: red;
font-family: segoe ui;
font-size: 18px;
opacity: 00.54;
padding-bottom: 2px;
padding-right: 14px;
}
`
```

Copyright © 2001 -2024 Syncfusion Inc.

Keyboard support in Vue Rich text editor component

The editor has full keyboard accessibility that includes shortcuts to open and other actions with toolbar items, drop-down lists, and dialogs.

HTML formation shortcut key

You can use the following key shortcuts when the Rich Text Editor renders with HTML editMode.

Actions	Keyboard shortcuts
Toolbar focus	Alt + f10
Insert link	Ctrl + k
Insert image	Ctrl + Shift + i
Insert audio	Ctrl + Shift + a
Insert video	Ctrl + Alt + v
Insert table	Ctrl + Shift + e
Undo	Ctrl + z
Redo	Ctrl + y
Copy	Ctrl + c
Cut	Ctrl + x
Paste	Ctrl + v
Bold	Ctrl + b
Italic	Ctrl + i
Underline	Ctrl + u
Strikethrough	Ctrl + Shift + s
Uppercase	Ctrl + Shift + u
Lowercase	Ctrl + Shift + l
Superscript	Ctrl + Shift + =
Subscript	Ctrl + =
Indents	Ctrl +]
Outdents	Ctrl + [
HTML source	Ctrl + Shift + h
Fullscreen	Ctrl + Shift + f
Exit Fullscreen	Esc
Justify center	Ctrl + e

Copyright © 2001 -2024 Syncfusion Inc.

| Justify full | Ctrl + j |

| Justify left | Ctrl + l |

| Justify right | Ctrl + r |

| Clear format | Ctrl + Shift + r |

| Ordered list | Ctrl + Shift + o |

| Unordered list | Ctrl + Alt + o |

| Format Painter Copy| Alt + Shift + c |

| Format Painter Paste| Alt + Shift + v |

| Format Painter Escape | Esc |

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340"
:toolbarSettings="toolbarData">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, HtmlEditor, Image, QuickToolbar,
Link, FormatPainter } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        toolbarData: {
          items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
            'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
            'LowerCase', 'UpperCase', '|',
            'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
            'Outdent', 'Indent', '|',
            'CreateLink', 'Image', '|', 'FormatPainter', 'ClearFormat',
'Print',
            'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
        }
      }
    },
    mounted() {
      document.addEventListener('keyup', this.onKeyUp);
    },
    methods: {
      onKeyUp (e) {
        if (e.altKey && e.keyCode === 84) { /* t */
          // press alt+t to focus the component.
          this.$refs.defaultRTE.ej2Instances.focusIn();
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, HtmlEditor, Image, QuickToolbar, Link,
FormatPainter ]
    }
  }
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs21" %}

Markdown formation shortcut key

You can use the following key shortcuts when the Rich Text Editor renders with Markdown editMode.

Copyright © 2001 -2024 Syncfusion Inc.

| Actions | Keyboard shortcuts |

|---|---|

| Toolbar focus | Alt + f10 |

| Insert link | Ctrl + k |

| Insert image | Ctrl + Shift + i |

| Insert table | Ctrl + Shift + e |

| Undo | Ctrl + z |

| Redo | Ctrl + y |

| Copy | Ctrl + c |

| Cut | Ctrl + x |

| Paste | Ctrl + v |

| Bold | Ctrl + b |

| Italic | Ctrl + i |

| Strikethrough | Ctrl + Shift + s |

| Uppercase | Ctrl + Shift + u |

| Lowercase | Ctrl + Shift + l |

| Superscript | Ctrl + Shift + = |

| Subscript | Ctrl + = |

| Fullscreen | Ctrl + Shift + f |

| Ordered list | Ctrl + Shift + o |

| Unordered list | Ctrl + Alt + o |

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340"
:toolbarSettings="toolbarData" :valueTemplate="valueTemplate"
editorMode="Markdown">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, MarkdownEditor } from
'@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        toolbarData: {
          items: ['Bold', 'Italic', 'StrikeThrough', '|',
            'Formats', 'OrderedList', 'UnorderedList', '|',
            'CreateLink', 'Image', '|','Undo', 'Redo']
        },
        valueTemplate: `The sample is added to showcase **markdown
editing**.
  Type or edit the content and apply formatting to view markdown formatted
content.
  We can add our own custom formation syntax for the Markdown formation,
[sample link](https://ej2.syncfusion.com/home/).
  The third-party library <b>Marked</b> is used in this sample to convert
markdown into HTML content`
      }
    },
    mounted() {
      document.addEventListener('keyup', this.onKeyUp);
    },
    methods: {
      onKeyUp (e) {
        if (e.altKey && e.keyCode === 84) { /* t */
          // press alt+t to focus the component.
          this.$refs.defaultRTE.ej2Instances.focusIn();
        }
      }
    },
    provide: {
```

```
      richtexteditor: [Toolbar, Link, Image, MarkdownEditor]
    }
  }
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs22" %}

Custom key config

Customize the key config for the keyboard interaction of Rich Text Editor, using the keyConfig property.

In the following sample, customize the cut, copy, paste toolbar action with ctrl+1, ctrl+2, ctrl+3, respectively.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="340"
:toolbarSettings="toolbarData" :keyConfigs="keyConfig">
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
```

```
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        toolbarData: {
          items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
            'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
            'LowerCase', 'UpperCase', '|',
            'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
            'Outdent', 'Indent', '|',
            'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
            'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
        },
        keyConfig: {
          'copy': 'ctrl+1',
          'cut': 'ctrl+2',
          'paste': 'ctrl+3'
        }
      }
    },
    mounted() {
      document.addEventListener('keyup', this.onKeyUp);
    },
    methods: {
      onKeyUp (e) {
        if (e.altKey && e.keyCode === 84) { /* t */
          // press alt+t to focus the component.
          this.$refs.defaultRTE.ej2Instances.focusIn();
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs23" %}

See Also

- Globalization
- Accessibility
- How to customize the toolbar items shortcut key
- How to customize the saving operation

Accessibility in Vue Rich text editor component

The Rich Text Editor component has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties. This component is characterized by complete ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The accessibility compliance for the Rich Text Editor component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

ARIA attributes

The toolbar of Rich Text Editor, assigned the role of Toolbar and has the following list of ARIA attributes.

| **Roles and Attributes** | **Functionalities** |

| --- | --- |

| role="toolbar" | This attribute added to the toolbar element describes the actual role of the element. |

| aria-orientation | Indicates the toolbar orientation. Default value is horizontal. |

| aria-haspopup | Indicates the popup mode of the toolbar. The default value is false. When popup mode is enabled, attribute value has to be changed to true. |

| aria-disabled | Indicates the disabled state of the toolbar. |

| aria-owns | Identifies an element to define a visual, functional, or contextual parent/child relationship between DOM elements when the DOM hierarchy cannot represent the relationship. In the Rich Text Editor, the attribute contains the ID of the Rich Text Editor to indicate the popup as a child element. |

For further details of toolbar ARIA attributes, refer the accessibility of Toolbar documentation.

The Rich Text Editor element is assigned the role of application.

| **Roles and Attributes** | **Functionalities** |

| --- | --- |

| role="application" | This attribute added to the Rich Text Editor element describes the actual role of the element. |

| aria-disabled | Indicates the disabled state of the Rich Text Editor. |

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :height="350"
:iframeSettings="iframeData">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
      <li>
        <p>Provides HTML view to edit the source directly for
developers.</p>
      </li>
      <li>
        <p>Supports third-party library integration.</p>
      </li>
      <li>
        <p>Allows preview of modified content before saving it.</p>
      </li>
      <li>
        <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        iframeData: {
          enable: true
        }
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs1" %}

Keyboard interaction

The Rich Text Editor component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Rich Text Editor component.

HTML formation shortcut key

You can use the following key shortcuts when the Rich Text Editor renders with HTML edit mode.

| Actions | Keyboard shortcuts |

|---|---|

| Toolbar focus | Alt + f10 |

| Insert link | Ctrl + k |

| Insert image | Ctrl + Shift + i |

| Insert table | Ctrl + Shift + e |

| Undo | Ctrl + z |

| Redo | Ctrl + y |

| Copy | Ctrl + c |

| Cut | Ctrl + x |

| Paste| Ctrl + v |

| Bold| Ctrl + b |

| Italic| Ctrl + i |

| Underline| Ctrl + u |

| Strikethrough| Ctrl + Shift + s |

| Uppercase| Ctrl + Shift + u |

| Lowercase| Ctrl + Shift + l |

| Superscript| Ctrl + Shift + = |

| Subscript| Ctrl + = |

| Indents| Ctrl +] |

Copyright © 2001 -2024 Syncfusion Inc.

| Outdents| Ctrl + [|

| HTML source | Ctrl + Shift + h |

| Fullscreen| Ctrl + Shift + f |

| Exit Fullscreen| Esc |

| Justify center| Ctrl + e |

| Justify full | Ctrl + j |

| Justify left | Ctrl + l |

| Justify right | Ctrl + r |

| Clear format | Ctrl + Shift + r |

| Ordered list | Ctrl + Shift + o |

| Unordered list | Ctrl + Alt + o |

| Format Painter Copy| Alt + Shift + c |

| Format Painter Paste| Alt + Shift + v |

| Format Painter Escape | Esc |

Markdown formation shortcut key

You can use the following key shortcuts when the Rich Text Editor renders with Markdown edit mode

| Actions | Keyboard shortcuts |

|---|---|

| Toolbar focus| Alt + f10 |

| Insert link| Ctrl + k |

| Insert image| Ctrl + Shift + i |

| Insert table| Ctrl + Shift + e |

| Undo| Ctrl + z |

| Redo| Ctrl + y |

| Copy| Ctrl + c |

| Cut| Ctrl + x |

| Paste| Ctrl + v |

| Bold| Ctrl + b |

| Italic| Ctrl + i |

| Strikethrough| Ctrl + Shift + s |

| Uppercase| Ctrl + Shift + u |

| Lowercase| Ctrl + Shift + l |

Copyright © 2001 -2024 Syncfusion Inc.

| Superscript| Ctrl + Shift + = |

| Subscript| Ctrl + = |

| Fullscreen| Ctrl + Shift + f |

| Ordered list| Ctrl + Shift + o |

| Unordered list| Ctrl + Alt + o |

Ensuring accessibility

The Rich Text Editor component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Rich Text Editor component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Rich Text Editor component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/rich-text-editor.html" %}

See also

- Accessibility in Syncfusion Vue components

How To

Add google fonts in Vue Rich text editor component

To use web fonts in Rich Text Editor, it is not needed for the web fonts to be present in local machine. To add the web fonts to Rich Text Editor, we need to refer the web font links and add the font names in the fontFamily property.

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj" :fontFamily="fontFamily"
:toolbarSettings="toolbarSettings" ><p>The Rich Text Editor component is
WYSIWYG ("what you see is what you get") editor that provides the best user
experience to create and update the content. Users can format their content
using standard toolbar commands.</p>
        <p><b>Key features:</b></p>
          <ul>
            <li><p>Provides IFRAME and DIV modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
          </ul></ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
```

```
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Count, Image, HtmlEditor,
QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
       fontFamily: {
        items:[
          {text: "Segoe UI", value: "Segoe UI", class: "e-segoe-ui",
command: "Font", subCommand: "FontName"},
          {text: "Roboto", value: "Roboto",  command: "Font", subCommand:
"FontName"}, // here font is added
          {text: "Great vibes", value: "Great Vibes,cursive",  command:
"Font", subCommand: "FontName"}, // here font is added
          {text: "Noto Sans", value: "Noto Sans",  command: "Font",
subCommand: "FontName"},
          {text: "Impact", value: "Impact,Charcoal,sans-serif", class: "e-
impact", command: "Font", subCommand: "FontName"},
          {text: "Tahoma", value: "Tahoma,Geneva,sans-serif", class: "e-
tahoma", command: "Font", subCommand: "FontName"},
        ]
      },
        toolbarSettings: {
            items: ['Bold', 'Italic', 'Underline', 'StrikeThrough','|',
                'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
                'LowerCase', 'UpperCase', '|',
                'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
                'Outdent', 'Indent', '|',
                'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
                'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
        },
    }
     },
    provide:{
        richtexteditor: [Toolbar, Link, Count, Image, HtmlEditor,
QuickToolbar]
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs6" %}

Copyright © 2001 -2024 Syncfusion Inc.

The below font style links are referred in the page.

`ts

<link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Roboto">

<link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Great+Vibes">

`

In the above sample, you can see that we have added two Google web fonts (Roboto and Great vibes) to Rich Text Editor.

Change default font family in Vue Rich text editor component

By using default property, you can change the default font-family of the Rich Text Editor. To change the font-family of the Rich Text Editor content while loading, we need to give the font-family in the style section with the help of cssClass property.

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj" :fontFamily="fontFamily"
:cssClass="cssClass" :toolbarSettings="toolbarSettings" ><p>The Rich Text
Editor component is WYSIWYG ("what you see is what you get") editor that
provides the best user experience to create and update the content. Users
can format their content using standard toolbar commands.</p>
        <p><b>Key features:</b></p>
          <ul>
            <li><p>Provides IFRAME and DIV modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
          </ul></ejs-richtexteditor>
        </div>
    </div>
</div>
<style>
 .customClass .e-rte-content .e-content {
            /* to get the desired font on intially*/
            font-family: "Noto Sans";
        }
        </style>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Count, Image, HtmlEditor,
QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
       fontFamily: {
```

```
          default:"Noto Sans", // to define default font-family
        items:[
          {text: "Segoe UI", value: "Segoe UI", class: "e-segoe-ui",
command: "Font", subCommand: "FontName"},
          {text: "Noto Sans", value: "Noto Sans",  command: "Font",
subCommand: "FontName"},
          {text: "Impact", value: "Impact,Charcoal,sans-serif", class: "e-
impact", command: "Font", subCommand: "FontName"},
          {text: "Tahoma", value: "Tahoma,Geneva,sans-serif", class: "e-
tahoma", command: "Font", subCommand: "FontName"},
        ]
      },
        toolbarSettings: {
            items: ['Bold', 'Italic', 'Underline', 'StrikeThrough','|',
                'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
                'LowerCase', 'UpperCase', '|',
                'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
                'Outdent', 'Indent', '|',
                'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
                'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
        },
        cssClass: "customClass",
    }
     },
    provide:{
        richtexteditor: [Toolbar, Link, Count, Image, HtmlEditor,
QuickToolbar]
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs7" %}

Check image size in Vue Rich text editor component

By using the Rich text editor's `imageUploading` event, you can get the image size before uploading and restrict the image to upload, when the given image size is greater than the allowed size.

In the following, we have validated the image size before uploading and determined whether the image has been uploaded or not.

APP.VUE

```
<template>
```

```
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor :imageUploading="onImageUploading"
:insertImageSettings="insertImageSettings"
:toolbarSettings="toolbarSettings" >
          </ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
    import Vue from "vue";
    import { RichTextEditorPlugin, Toolbar, Link, Count, Image, HtmlEditor,
QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
    Vue.use(RichTextEditorPlugin);
    export default {
        data: function() {
            return {
            toolbarSettings: {
                items: ['Bold', 'Italic', 'Underline', 'StrikeThrough','|',
                    'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
                    'LowerCase', 'UpperCase', '|',
                    'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
                    'Outdent', 'Indent', '|',
                    'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
                    'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
            },
            insertImageSettings: {
                saveUrl:
"https://aspnetmvc.syncfusion.com/services/api/uploadbox/Save",
                path: "../Images/"
            },
        }
        },
        methods: {
            onImageUploading: function(args){
                console.log("file is uploading");
                var imgSize = 500000;
                var sizeInBytes = args.fileData.size;
                if ( imgSize < sizeInBytes ) {
                args.cancel = true;
            }
            }
            },
            provide:{
                richtexteditor: [Toolbar, Link, Count, Image, HtmlEditor,
QuickToolbar]
            }
        }
    }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
```

```
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs14" %}

Customize placeholder style in Vue Rich text editor component

By using e-rte-placeholder class, you can customize the placeholder style.

APP.VUE

```
<template>
  <div>
    <ejs-richtexteditor ref="defaultRTE" :placeholder="placeholder" >
    </ejs-richtexteditor>
  </div>
  <style>
        .e-richtexteditor .e-rte-placeholder {
            font-family: monospace;
            color: deeppink;
        }
    </style>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
          placeholder: "Type Something",
      }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
   @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/getting-started-cs15" %}

Customize shortcut keys in Vue Rich text editor component

It can be achieved by using formatter property. We need to create customformatterModel to configure the keyConfig using IHtmlFormatterModel class and assign the same to the formatter property. Here, ctrl+q is configured to open the Insert Hyperlink dialog.

APP.VUE

```
<template>
  <ejs-richtexteditor ref="defaultRTE" :placeholder="placeholder"
:formatter="formatter" >
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
</template>
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, Toolbar, Link, Image, HtmlEditor,
QuickToolbar, IHtmlFormatterModel, HTMLFormatter } from '@syncfusion/ej2-
vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
    var customHTMLModel = {
  keyConfig: {
'insert-link': 'ctrl+q', // confite the desired key
}
}
      return {
        placeholder: "Type Something",
        formatter: new HTMLFormatter(customHTMLModel), // to configure
custom key
    }
    },
```

```
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs8" %}

We need to import IHtmlFormatterModel and HTMLFormatter to configure the shortcut key.

Set the cursor at the specific range in Vue Rich text editor component

This can be achieved by using setRange method in the Rich Text Editor using NodeSelection instance. In this below sample, we have passed the text node (specific location in Rich Text Editor content) in setStart method and passed the range in setRange method of Rich Text Editor.

APP.VUE

```
<template>
<div>
  <ejs-richtexteditor ref="defaultRTE" :placeholder="placeholder" >
    <p>The Rich Text Editor component is WYSIWYG ("what you see is what you
get") editor that provides the best user experience to create and update the
content. Users can format their content using standard toolbar commands.</p>
    <p id="key"><b>Key features:</b></p>
    <ul>
      <li>
        <p>Provides IFRAME and DIV modes</p>
      </li>
      <li>
        <p>Capable of handling markdown editing.</p>
      </li>
      <li>
        <p>Contains a modular library to load the necessary functionality on
demand.</p>
      </li>
      <li>
        <p>Provides a fully customizable toolbar.</p>
      </li>
    </ul>
  </ejs-richtexteditor>
  <button v-on:click="onClick" class="e-btn">Set Cursor Position</button>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
  import Vue from 'vue';
  import { RichTextEditorPlugin, NodeSelection, Toolbar, Link, Image,
HtmlEditor, QuickToolbar } from '@syncfusion/ej2-vue-richtexteditor';
  Vue.use(RichTextEditorPlugin);
  export default {
    data() {
      return {
        placeholder: "Type Something",
    }
    },
   methods: {
     onClick: function(event){
       var instance = this.$refs.defaultRTE.$el.ej2_instances[0];
       let element: Element=
instance.contentModule.getDocument().getElementById("key");
       let selectioncursor: NodeSelection = new NodeSelection();
       let range: Range = document.createRange();
       range.setStart(element, 1); // to set the range
       selectioncursor.setRange(document, range); // to set the cursor
     }
    },
    provide: {
      richtexteditor: [Toolbar, Link, Image, HtmlEditor, QuickToolbar]
    }
  }
</script>
<style>
   @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs10" %}

Update value in Vue Rich text editor component

To achieve this, we need to bind the keydown event to the Rich Text Editor content and capture the ctrl + s key press using its keyCode.

In the keydown event handler, the updateValue method is called to update the value property and then we can save the content in the required database using the same.

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
```

```
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj" :toolbarSettings="toolbarSettings"
:created="onCreate"><p>The Rich Text Editor component is WYSIWYG ("what you
see is what you get") editor that provides the best user experience to
create and update the content. Users can format their content using standard
toolbar commands.</p>
        <p><b>Key features:</b></p>
          <ul>
            <li><p>Provides IFRAME and DIV modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
          </ul></ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, HtmlEditor } from "@syncfusion/ej2-
vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
        toolbarSettings: {
            type: 'MultiRow',
            items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
            'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
            'LowerCase', 'UpperCase', '|',
            'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
            'Outdent', 'Indent', '|',
            'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
            'SourceCode', 'FullScreen', '|', 'Undo', 'Redo'
          ]
        },
        }
    },
    methods: {
    onCreate: function(){
      var instance = this.$refs.rteObj.$el.ej2_instances[0];

instance.contentModule.getDocument().addEventListener("keydown",function(e:
any):void{
            if(e.key === 's' && e.ctrlKey===true){
                  e.preventDefault(); // to prevent default ctrl+s action
                  instance.updateValue(); // to update the value after
editing
                  let value: any= instance.value; // you can get the RTE
content to save in the desired database
            }
      });
    }
    },
    provide:{
```

```
        richtexteditor:[Toolbar, HtmlEditor]
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs11" %}

Rename images in server in Vue Rich text editor component

By using the insertImageSettings property, you can specify the server handler to upload the selected image. Then you can bind the imageUploadSuccess event, to receive the modified file name from the server and update it in the Rich Text Editor's insert image dialog.

`

<template>

<ejs-richtexteditor ref="rteObj" :imageUploadSuccess="onImageUploadSuccess" :insertImageSettings="insertImageSettings" :toolbarSettings="toolbarSettings" > The Rich Text Editor component is WYSIWYG ("what you see is what you get") editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.

Key features:

 Provides IFRAME and DIV modes

 Capable of handling markdown editing.

 Contains a modular library to load the necessary functionality on demand.

 Provides a fully customizable toolbar.

</ejs-richtexteditor>

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Count, Image, HtmlEditor, QuickToolbar } from "@syncfusion/ej2-vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
data: function() {
return {
toolbarSettings: {
items: ['Bold', 'Italic', 'Underline', 'StrikeThrough','|',
'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
'LowerCase', 'UpperCase', '|',
'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
'Outdent', 'Indent', '|',
'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
},
insertImageSettings: {
saveUrl: "[SERVICE*HOSTED*PATH]/api/uploadbox/Rename",
path: "../Images/"
},
}
},
methods: {
onImageUploadSuccess: function(args){
if (args.e.currentTarget.getResponseHeader('name') != null) {
args.file.name = args.e.currentTarget.getResponseHeader('name');
let filename: any = document.querySelectorAll(".e-file-name")[0];
filename.innerHTML = args.file.name.replace(document.querySelectorAll(".e-file-type")[0].innerHTML, '');
filename.title = args.file.name;
```

```
}
},
provide:{
richtexteditor: [Toolbar, Link, Count, Image, HtmlEditor, QuickToolbar]
},
}
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-richtexteditor/styles/material.css";
</style>
`
```

To configure the server-side handler, refer the below code.

```c#
[AcceptVerbs("Post")]
public void Rename()
{
try
{
var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
imageFile = httpPostedFile.FileName;
if (httpPostedFile != null)
{
var fileSave = System.Web.HttpContext.Current.Server.MapPath("~/Images");
if (!Directory.Exists(fileSave))
{
```

```
Directory.CreateDirectory(fileSave);
}
var fileName = Path.GetFileName(httpPostedFile.FileName);
var fileSavePath = Path.Combine(fileSave, fileName);
while (System.IO.File.Exists(fileSavePath))
{
imageFile = "rteImage" + x + "-" + fileName;
fileSavePath = Path.Combine(fileSave, imageFile);
x++;
}
if (!System.IO.File.Exists(fileSavePath))
{
httpPostedFile.SaveAs(fileSavePath);
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.Headers.Add("name", imageFile);
Response.ContentType = "application/json; charset=utf-8";
Response.StatusDescription = "File uploaded succesfully";
Response.End();
}
}
}
catch (Exception e)
{
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusCode = 204;
Response.Status = "204 No Content";
Response.StatusDescription = e.Message;
Response.End();
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

`

File attachment in Vue Rich text editor component

The Rich Text Editor allows you to attach a file based on the file upload. You can attach your files using the file upload or drag-and-drop from your local path. When the file upload gets success, the attachment link inserts into the content.

In the below sample, configure the saveUrl and path properties to achieve file attachments.

1. saveUrl: Provides service URL to save the files.
2. path: Specifies the location to store the image.

The following sample illustrates how to attach a file in the Rich Text Editor.

`

```
<template>
<div>
<div class="control-section">
<div class="sample-container">
<div class="default-section">
<ejs-richtexteditor ref="rteObj" :insertImageSettings="insertImageSettings">
<p>The Rich Text Editor component is WYSIWYG ("what you see is what you get") editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul>
<li><p>Provides IFRAME and DIV modes</p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
</ul>
</ejs-richtexteditor>
<ejs-uploader ref="uploadObj" :asyncSettings= "path"
:dropArea = "dropElement" :success= "onImageUploadSuccess" ></ejs-uploader>
</div>
</div>
</div>
</div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, Link, Image, Count, HtmlEditor, QuickToolbar, Table
,NodeSelection} from "@syncfusion/ej2-vue-richtexteditor";
import {UploaderPlugin} from "@syncfusion/ej2-vue-inputs";
Vue.use(RichTextEditorPlugin);
Vue.use(UploaderPlugin);
export default Vue.extend({
data: function() {
return {
selection: new NodeSelection(),
range: null,
dropElement: '.e-richtexteditor',
saveSelection : null,
insertImageSettings: {
saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/Save",
path: "../Files/"
},
path:  {
saveUrl: '[SERVICEHOSTEDPATH]/api/uploadbox/Save'
},
};
},
methods: {
onImageUploadSuccess: function(args){
this.$refs.rteObj.ej2Instances.contentModule.getEditPanel().focus();
this.range = this.selection.getRange(document);
this.saveSelection = this.selection.save(this.range, document);
var fileUrl = document.URL + this.$refs.rteObj.ej2Instances.insertImageSettings.path + args.file.name;
this.$refs.rteObj.ej2Instances.executeCommand('createLink', { url: fileUrl, text: fileUrl, selection:
this.saveSelection });
if (this.$refs.rteObj.ej2Instances.formatter.getUndoRedoStack().length === 0) {
this.$refs.rteObj.ej2Instances.formatter.saveData();
}
```

```
saveSelection.restore();
this.$refs.rteObj.ej2Instances.executeCommand('createLink', { url: fileUrl, text: fileUrl, selection: saveSelection });
this.$refs.rteObj.ej2Instances.formatter.saveData();
this.$refs.rteObj.ej2Instances.formatter.enableUndo(this.$refs.rteObj.ej2Instances);
this.$refs.uploadObj.ej2Instances.clearAll();
}
},
provide:{
richtexteditor:[Toolbar, Link, Image, Count, HtmlEditor, QuickToolbar, Table]
}
});
</script>
```

To config server-side handler, refer the below code.

```c#
int x = 0;
string file;
[AcceptVerbs("Post")]
public void Save()
{
try
{
var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
file = httpPostedFile.FileName;
if (httpPostedFile != null)
{
Console.WriteLine(System.Web.HttpContext.Current.Server.MapPath("~/Files"));
var fileSave = System.Web.HttpContext.Current.Server.MapPath("~/Files");
if (!Directory.Exists(fileSave))
{
Directory.CreateDirectory(fileSave);
}
```

```
var fileName = Path.GetFileName(httpPostedFile.FileName);
var fileSavePath = Path.Combine(fileSave, fileName);
while (System.IO.File.Exists(fileSavePath))
{
file = "rte" + x + "-" + fileName;
fileSavePath = Path.Combine(fileSave, file);
x++;
}
if (!System.IO.File.Exists(fileSavePath))
{
httpPostedFile.SaveAs(fileSavePath);
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.Headers.Add("name", file);
Response.ContentType = "application/json; charset=utf-8";
Response.StatusDescription = "File uploaded succesfully";
Response.Headers.Add("url", fileSavePath);
Response.End();
}
}
}
catch (Exception e)
{
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusCode = 204;
Response.Status = "204 No Content";
Response.StatusDescription = e.Message;
Response.End();
}
}
`
```

Format code block in Vue Rich text editor component

You can configure code block formatting as a separate toolbar button by adding the **InsertCode** keyword within the toolbarSettings items property.

The InsertCode button has a toggle state to apply code block formatting to the editor and remove code block formatting from the editor.

The following sample demonstrates how to config the InsertCode button in toolbar and set the background color to “pre” tag for highlighting the code block.

APP.VUE

```
<template>
<div>
<div class="control-section">
    <div class="sample-container">
        <div class="default-section">
        <ejs-richtexteditor ref="rteObj"
:toolbarSettings="toolbarSettings"><p>The Rich Text Editor component is
WYSIWYG ("what you see is what you get") editor that provides the best user
experience to create and update the content. Users can format their content
using standard toolbar commands.</p>
        <p><b>Key features:</b></p>
          <ul>
            <li><p>Provides IFRAME and DIV modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
          </ul></ejs-richtexteditor>
        </div>
    </div>
</div>
</div>
</template>
<style>
  .e-richtexteditor .e-rte-content .e-content pre {
    padding: 10px;
    background: #F4F5F7;
  }
</style>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Toolbar, HtmlEditor } from "@syncfusion/ej2-
vue-richtexteditor";
Vue.use(RichTextEditorPlugin);
export default {
     data: function() {
        return {
          toolbarSettings: {
          items: ['InsertCode'] },
        }
    },
    provide:{
        richtexteditor:[Toolbar, HtmlEditor]
    }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-lists/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
richtexteditor/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/rich-text-editor/toolbar-cs9" %}

Schedule

Getting Started with the Vue Schedule Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLILink to the Video and integrating the Syncfusion Vue Schedule component.

Check out this video to see how to develop a Vue Scheduler application:

Prerequisites

System requirements for Syncfusion Vue UI components

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

vue create quickstart

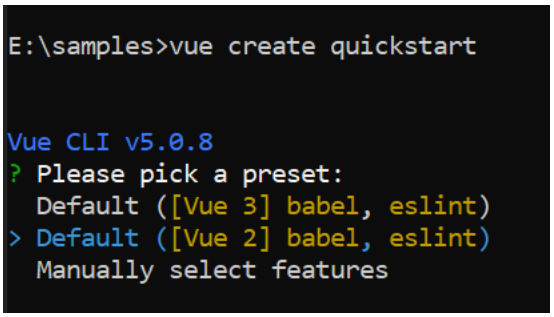
cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.

Copyright © 2001 -2024 Syncfusion Inc.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Schedule component as an example. Install the @syncfusion/ej2-vue-schedule package by running the following command:

`bash

npm install @syncfusion/ej2-vue-schedule --save

`

or

`bash

yarn add @syncfusion/ej2-vue-schedule

`

The **--save** will instruct NPM to include the Scheduler package inside of the dependencies section of the package.json.

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Schedule component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '/node_modules/@syncfusion/ej2-base/styles/material.css';
@import '/node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '/node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '/node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '/node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '/node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '/node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '/node_modules/@syncfusion/ej2-vue-schedule/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Schedule component using:

1\. First, import and register the Schedule component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { ScheduleComponent } from '@syncfusion/ej2-vue-schedule';
</script>
```

2\. In the template section, define the Schedule component property.

~/SRC/APP.VUE

```
<template>
<div id='app'>
<ejs-schedule ></ejs-schedule>
</div>
</template>
```

Module injection

The crucial step on creating a Schedule with specific views, is to inject the required modules. The modules that are available with common Schedule basic functionality are as follows.

- **Day** - Inject this module for displaying day view.
- **Week** - Inject this module for displaying week view.
- **WorkWeek** - Inject this module for displaying work week view.
- **Month** - Inject this module for displaying month view.
- **Agenda** - Inject this module for displaying agenda view.
- **MonthAgenda** - Inject this module for displaying month agenda view.
- **TimelineViews** - Inject this module for displaying timeline day, timeline week, timeline work week view.
- **TimelineMonth** - Inject this module for displaying timeline month view.

These modules should be injected into the Schedule using the provide method within the app.vue file as shown below. On doing so, only the injected views will be loaded and displayed on the Schedule.

[src/app/app.vue]

`html

<template>

<ejs-schedule ></ejs-schedule>

</template>

Copyright © 2001 -2024 Syncfusion Inc.

<script>

import { ScheduleComponent, Day, Week, WorkWeek, Month, Agenda } from '@syncfusion/ej2-vue-schedule';

export default {

components: {

'ejs-schedule': ScheduleComponent

},

provide: {

schedule: [Day, Week, WorkWeek, Month, Agenda]

}

}

</script>

`

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

The output will display the empty Scheduler.

Populating appointments

To populate the empty Scheduler with appointments, define either the local JSON data or remote data through the dataSource property available within the eventSettings option. To define any appointments, start and end time fields are mandatory. In the following example, you can see the appointment defined with default fields such as Id, Subject, StartTime and EndTime.

`html

<template>

<ejs-schedule height='550px' :selectedDate='selectedDate' :eventSettings='eventSettings'></ejs-schedule>

</template>

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import { ScheduleComponent, Day, Week, WorkWeek, Month, Agenda } from '@syncfusion/ej2-vue-schedule';
export default {
components: {
'ejs-schedule': ScheduleComponent
},
data () {
return {
selectedDate: new Date(2018, 1, 15),
eventSettings: {
dataSource: [{
Id: 1,
Subject: 'Meeting',
StartTime: new Date(2018, 1, 15, 10, 0),
EndTime: new Date(2018, 1, 15, 12, 30)
}]
}
}
},
provide: {
schedule: [Day, Week, WorkWeek, Month, Agenda]
}
}
</script>
```

You can also provide different names to these default fields, for which the custom names of those fields must be mapped appropriately within fields property as shown below.

```
`html
<template>
<div id='app'>
<ejs-schedule height='550px' :selectedDate='selectedDate' :eventSettings='eventSettings'></ejs-schedule>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import { ScheduleComponent, Day, Week, WorkWeek, Month, Agenda } from '@syncfusion/ej2-vue-schedule';
let data = [{
Id: 2,
EventName: 'Meeting',
StartTime: new Date(2018, 1, 15, 10, 0),
EndTime: new Date(2018, 1, 15, 12, 30),
IsAllDay: false
}];
export default {
components: {
'ejs-schedule': ScheduleComponent
},
data () {
return {
selectedDate: new Date(2018, 1, 15),
eventSettings: {
dataSource: data,
fields: {
id: 'Id',
subject: { name: 'EventName' },
isAllDay: { name: 'IsAllDay' },
startTime: { name: 'StartTime' },
endTime: { name: 'EndTime' }
}
}
}
},
provide: {
schedule: [Day, Week, WorkWeek, Month, Agenda]
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
```

The other fields available in Scheduler can be referred from here.

Setting date

Scheduler usually displays the system date as its current date. To change the current date of Scheduler with specific date, define the `selectedDate property.

[src/app/app.vue]

```
`html
<template>
<div id='app'>
<ejs-schedule height='550px' :selectedDate='selectedDate'></ejs-schedule>
</div>
</template>
<script>
import { ScheduleComponent, Day, Week, WorkWeek, Month, Agenda } from '@syncfusion/ej2-vue-schedule';
export default {
components: {
'ejs-schedule': ScheduleComponent
},
data (){
return {
selectedDate: new Date(2018, 1, 15)
}
},
provide: {
schedule: [Day, Week, WorkWeek, Month, Agenda]
}
}
</script>
`
```

Copyright © 2001 -2024 Syncfusion Inc.

Setting view

Scheduler displays week view by default. To change the current view, define the applicable view name to the currentView property. The applicable view names are,

- Day
- Week
- WorkWeek
- Month
- Year
- Agenda
- MonthAgenda
- TimelineDay
- TimelineWeek
- TimelineWorkWeek
- TimelineMonth
- TimelineYear

```
`html

<template>

<div id='app'>

<ejs-schedule height='550px' :selectedDate='selectedDate' :currentView='currentView'>

</ejs-schedule>

</div>

</template>

<script>

import { ScheduleComponent, Day, WorkWeek, Agenda, Month, Week } from '@syncfusion/ej2-vue-schedule';

export default {

components: {

'ejs-schedule': ScheduleComponent

},

data () {

return {

selectedDate: new Date(2018, 1, 15),

currentView: 'Month',

}

},

provide: {

schedule: [Day, WorkWeek, Agenda, Month, Week]
```

Copyright © 2001 -2024 Syncfusion Inc.

}

}

</script>

`

Individual view customization

Each individual Scheduler views can be customized with its own options such as setting different start and end hour on Week and Work Week views, whereas hiding the weekend days on Month view alone.

This can be achieved by defining views property to accept the array of object type, where each object depicts the individual view customization.

The output will display the Scheduler with the specified view configuration.

~/SRC/APP.VUE

```
<template>
  <div id='app'>
      <ejs-schedule height='550px' width='100%' :selectedDate='selectedDate'
      :eventSettings= 'eventSettings'>
        <e-views>
          <e-view option='Week' startHour='07:00' endHour='15:00' ></e-view>
          <e-view option='WorkWeek' startHour='10:00' endHour='18:00' ></e-
view>
          <e-view option='Month' showWeekend=false></e-view>
        </e-views>
      </ejs-schedule>
  </div>
</template>
<script>
  import { scheduleData } from './datasource.js';
  import { extend } from '@syncfusion/ej2-base';
  import { ScheduleComponent, ViewsDirective, ViewDirective,  WorkWeek,
Month, Week } from '@syncfusion/ej2-vue-schedule';
  export default {
    components: {
      'ejs-schedule': ScheduleComponent,
      'e-view': ViewDirective,
      'e-views': ViewsDirective
    },
    data () {
      return {
        eventSettings: { dataSource: extend([], scheduleData, null, true) },
        selectedDate: new Date(2018, 1, 15)
      }
    },
    provide: {
      schedule: [ WorkWeek, Month, Week]
    }
  }
</script>
<style>
@import '/node_modules/@syncfusion/ej2-base/styles/material.css';
@import '/node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '/node_modules/@syncfusion/ej2-calendars/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '/node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '/node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '/node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '/node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '/node_modules/@syncfusion/ej2-vue-schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs1" %}

You can also explore our Vue Scheduler example that shows how to use the toolbar buttons to play with Scheduler functionalities.

Getting Started with Syncfusion Scheduler Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Scheduler component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will creates a Vue 3 project.

Copyright © 2001 -2024 Syncfusion Inc.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion Scheduler component to the project.

Adding Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

Copyright © 2001 -2024 Syncfusion Inc.

This article uses the Vue Scheduler component as an example. To use the Vue Scheduler component in the project, the @syncfusion/ej2-vue-schedule package needs to be installed using the following command

`bash

npm install @syncfusion/ej2-vue-schedule --save

`

or

`bash

yarn add @syncfusion/ej2-vue-schedule

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Scheduler component and its dependents were imported into the <style> section of the **src/App.vue** file.

`html

<style>

@import '../node_modules/@syncfusion/ej2-base/styles/material.css';

@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';

@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';

@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';

@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';

@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';

@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';

@import '../node_modules/@syncfusion/ej2-vue-schedule/styles/material.css';

</style>

`

Note: Scheduler component use other Syncfusion components too, the dependent component's CSS references need to be added for using all the Scheduler functionalities.

Add Syncfusion Vue component

Follow the below steps to add the Vue Scheduler component using Composition API or Options API:

1.First, import and register the Scheduler component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

Copyright © 2001 -2024 Syncfusion Inc.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { provide } from "vue";
import {
ScheduleComponent as EjsSchedule, ViewsDirective as EViews, ViewDirective as
EView,
ResourcesDirective as EResources, ResourceDirective as EResource,
Day, Week, WorkWeek, Month, Agenda
} from "@syncfusion/ej2-vue-schedule";
provide('schedule', [Day, Week, WorkWeek, Month, Agenda]);
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import {
Day,
Week,
WorkWeek,
Month,
Agenda,
MonthAgenda,
ScheduleComponent, ViewsDirective, ViewDirective
} from "@syncfusion/ej2-vue-schedule";
export default {
name: "App",
components: {
'ejs-schedule' : ScheduleComponent,
'e-views' : ViewsDirective,
'e-view' : ViewDirective,
'e-resources': ResourcesDirective,
'e-resource': ResourceDirective
}
}
</script>
```

2.In the template section, define the Scheduler component with the selectedDate, eventSettings properties and resources definition with dataSource property.

`html

<template>

<ejs-schedule height='550px' width='100%' :selectedDate='selectedDate'
:eventSettings='eventSettings'>

<e-views>

<e-view option='Day'></e-view>

<e-view option='Week'></e-view>

<e-view option='WorkWeek'></e-view>

<e-view option='Month'></e-view>

<e-view option='Agenda'></e-view>

Copyright © 2001 -2024 Syncfusion Inc.

</e-views>

<e-resources>

<e-resource field="OwnerId" title="Owner" name="Owners" :dataSource="ownerDataSource" textField="OwnerText"

idField="Id" colorField="OwnerColor">

</e-resource>

</e-resources>

</ejs-schedule>

</template>

`

3.Declare the values for the selectedDate, eventSettings and resources dataSource properties in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const selectedDate = new Date(2023, 7, 8);
const eventSettings = {
dataSource: [
{
Id: 1,
Subject: 'Surgery - Andrew',
EventType: 'Confirmed',
StartTime: new Date(2023, 7, 10, 9, 0),
EndTime: new Date(2023, 7, 10, 10, 0),
OwnerId: 2
},
{
Id: 2,
Subject: 'Consulting - John',
EventType: 'Confirmed',
StartTime: new Date(2023, 7, 9, 10, 0),
EndTime: new Date(2023, 7, 9, 11, 30),
OwnerId: 3
},
{
Id: 3,
Subject: 'Therapy - Robert',
EventType: 'Requested',
StartTime: new Date(2023, 7, 8, 11, 30),
EndTime: new Date(2023, 7, 8, 12, 30),
OwnerId: 1
}
]
};
const ownerDataSource = [
{ OwnerText: "Nancy", Id: 1, OwnerColor: "#ffaa00" },
{ OwnerText: "Steven", Id: 2, OwnerColor: "#f8a398" },
{ OwnerText: "Michael", Id: 3, OwnerColor: "#7499e1" }
];
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data: function () {
return {
selectedDate = new Date(2023, 7, 8),
eventSettings = {
dataSource: [
{
Id: 1,
Subject: 'Surgery - Andrew',
EventType: 'Confirmed',
StartTime: new Date(2023, 7, 10, 9, 0),
EndTime: new Date(2023, 7, 10, 10, 0),
OwnerId: 2
},
{
Id: 2,
Subject: 'Consulting - John',
EventType: 'Confirmed',
StartTime: new Date(2023, 7, 9, 10, 0),
EndTime: new Date(2023, 7, 9, 11, 30),
OwnerId: 3
},
{
Id: 3,
Subject: 'Therapy - Robert',
EventType: 'Requested',
StartTime: new Date(2023, 7, 8, 11, 30),
EndTime: new Date(2023, 7, 8, 12, 30),
OwnerId: 1
}
]
},
ownerDataSource =[
{ OwnerText: "Nancy", Id: 1, OwnerColor: "#ffaa00" },
{ OwnerText: "Steven", Id: 2, OwnerColor: "#f8a398" },
{ OwnerText: "Michael", Id: 3, OwnerColor: "#7499e1" }
]
}
},
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-schedule height='550px' width='100%' :selectedDate='selectedDate'
:eventSettings='eventSettings'>
<e-views>
<e-view option='Day'></e-view>
<e-view option='Week'></e-view>
<e-view option='WorkWeek'></e-view>
```

```
<e-view option='Month'></e-view>
<e-view option='Agenda'></e-view>
</e-views>
<e-resources>
<e-resource field="OwnerId" title="Owner" name="Owners"
:dataSource="ownerDataSource" textField="OwnerText"
idField="Id" colorField="OwnerColor">
</e-resource>
</e-resources>
</ejs-schedule>
</template>
<script setup>
import { provide } from "vue";
import {
ScheduleComponent as EjsSchedule, ViewsDirective as EViews, ViewDirective as
EView,
ResourcesDirective as EResources, ResourceDirective as EResource,
Day, Week, WorkWeek, Month, Agenda
} from "@syncfusion/ej2-vue-schedule";
provide('schedule', [Day, Week, WorkWeek, Month, Agenda]);
const selectedDate = new Date(2023, 7, 8);
const eventSettings = {
dataSource: [
{
Id: 1,
Subject: 'Surgery - Andrew',
EventType: 'Confirmed',
StartTime: new Date(2023, 7, 10, 9, 0),
EndTime: new Date(2023, 7, 10, 10, 0),
OwnerId: 2
},
{
Id: 2,
Subject: 'Consulting - John',
EventType: 'Confirmed',
StartTime: new Date(2023, 7, 9, 10, 0),
EndTime: new Date(2023, 7, 9, 11, 30),
OwnerId: 3
},
{
Id: 3,
Subject: 'Therapy - Robert',
EventType: 'Requested',
StartTime: new Date(2023, 7, 8, 11, 30),
EndTime: new Date(2023, 7, 8, 12, 30),
OwnerId: 1
}
]
};
const ownerDataSource = [
{ OwnerText: "Nancy", Id: 1, OwnerColor: "#ffaa00" },
{ OwnerText: "Steven", Id: 2, OwnerColor: "#f8a398" },
{ OwnerText: "Michael", Id: 3, OwnerColor: "#7499e1" }
];
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-vue-schedule/styles/material.css';
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-schedule height='550px' width='100%' :selectedDate='selectedDate'
:eventSettings='eventSettings'>
<e-views>
<e-view option='Day'></e-view>
<e-view option='Week'></e-view>
<e-view option='WorkWeek'></e-view>
<e-view option='Month'></e-view>
<e-view option='Agenda'></e-view>
</e-views>
<e-resources>
<e-resource field="OwnerId" title="Owner" name="Owners"
:dataSource="ownerDataSource" textField="OwnerText"
idField="Id" colorField="OwnerColor">
</e-resource>
</e-resources>
</ejs-schedule>
</template>
<script>
import {
Day,
Week,
WorkWeek,
Month,
Agenda,
MonthAgenda, ScheduleComponent, ViewsDirective, ViewDirective,
ResourcesDirective, ResourceDirective
} from "@syncfusion/ej2-vue-schedule";
export default {
name: "App",
components: {
'ejs-schedule': ScheduleComponent,
'e-views': ViewsDirective,
'e-view': ViewDirective,
'e-resources': ResourcesDirective,
'e-resource': ResourceDirective
},
data: function () {
return {
selectedDate = new Date(2023, 7, 8),
eventSettings = {
dataSource: [
{
Id: 1,
Subject: 'Surgery - Andrew',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
EventType: 'Confirmed',
StartTime: new Date(2023, 7, 10, 9, 0),
EndTime: new Date(2023, 7, 10, 10, 0),
OwnerId: 2
},
{
Id: 2,
Subject: 'Consulting - John',
EventType: 'Confirmed',
StartTime: new Date(2023, 7, 9, 10, 0),
EndTime: new Date(2023, 7, 9, 11, 30),
OwnerId: 3
},
{
Id: 3,
Subject: 'Therapy - Robert',
EventType: 'Requested',
StartTime: new Date(2023, 7, 8, 11, 30),
EndTime: new Date(2023, 7, 8, 12, 30),
OwnerId: 1
}
]
},
ownerDataSource =[
{ OwnerText: "Nancy", Id: 1, OwnerColor: "#ffaa00" },
{ OwnerText: "Steven", Id: 2, OwnerColor: "#f8a398" },
{ OwnerText: "Michael", Id: 3, OwnerColor: "#7499e1" }
]
}
},
provide: {
schedule: [Day, Week, WorkWeek, Month, Agenda, MonthAgenda]
}
};
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-vue-schedule/styles/material.css';
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

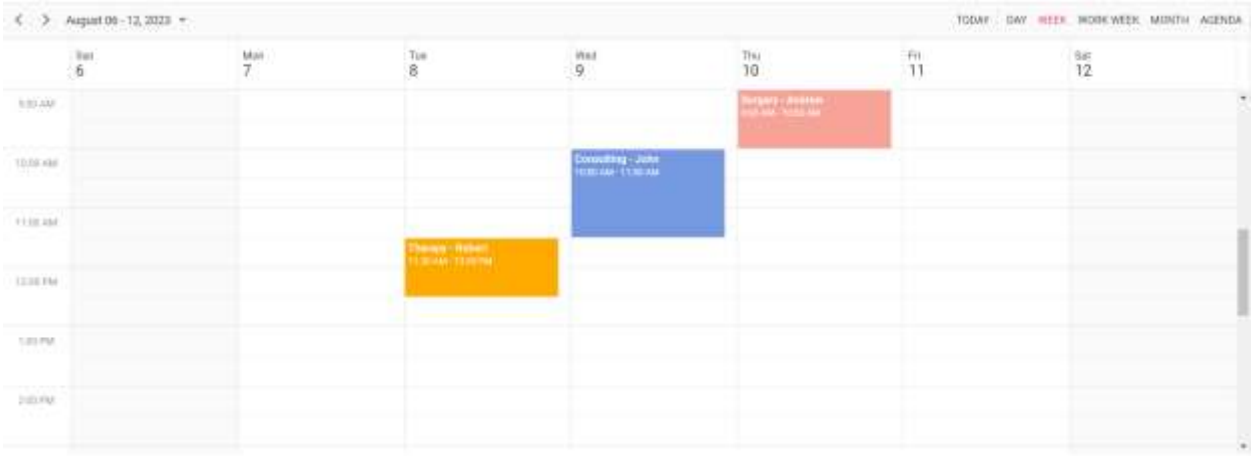
or

`bash

yarn run dev

`

The output will appear as follows:



Refer the sample Vue 3 using Composition API Scheduler getting started

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Module injection in Vue Schedule component

The crucial step on creating a Scheduler with required views, is to import and inject the required modules. The modules that are available on Scheduler to work with its related functionalities are as follows.

- Day - Inject this module to work with day view.
- Week - Inject this module to work with week view.
- WorkWeek - Inject this module to work with work week view.
- Month - Inject this module to work with month view.
- Year - Inject this module to work with year view.
- Agenda - Inject this module to work with agenda view.
- MonthAgenda - Inject this module to work with month agenda view.
- TimelineViews - Inject this module to work with timeline day, timeline week, and timeline work week views.
- TimelineMonth - Inject this module to work with timeline month view.
- TimelineYear - Inject this module to work with timeline year view.
- DragAndDrop - Inject this module to allow drag and drop of appointments on Scheduler.
- Resize - Inject this module for enabling the resize functionality of appointments on Scheduler.
- ExcelExport - Inject this module for exporting the Scheduler events data as excel file format.

Copyright © 2001 -2024 Syncfusion Inc.

- ICalendarExport - Inject this module for exporting the Scheduler events data to an ICS file.
- ICalendarImport - Inject this module for importing the Scheduler events data from an ICS file.

Module injection

The required modules should be injected into the Scheduler using the provide method of Scheduler within the app.vue file as shown below. On doing so, only the injected module functionalities will be loaded and can be worked with Scheduler.

[src/app.vue]

`html

provide: {

schedule: [Day, Week, WorkWeek, Month, Agenda, MonthAgenda]

},

`

Note: If a Scheduler currentView is set to any one of the available views without injecting that respective view module, then a script error will occur and the Scheduler will not render.

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Scheduler interactions in Vue Schedule component

The following table describes the Scheduler actions and also illustrates how those actions are carried out through mouse and touch interactions on Scheduler.

|Actions | Mouse interaction | Touch interaction |

|---|---|---|

| Single click or tap on cells | Single click on a cell to select a cell. | Single tapping on cells, will display a + icon on the cell. Tapping on it again will open the new event editor window. |

| Multiple cell selection | Single click on a cell and drag the selection to other cells to enable multiple cell selection. | No multiple cell selection is allowed using touch gestures. |

| Event selection | Single click on an event to select it. | Tap holding on events, select an event and opens a small popup at the top holding the options to edit or delete. The popup also displays the selected event's subject. |

| Multiple event selection and deletion | Pressing Ctrl key and altogether single clicking on multiple events one after the other will enable multiple event selection. Pressing Delete key after event selection will delete all the selected events. | Tap hold an event to select it, which opens a small popup at the top holding the options to edit or delete. As a continuation of this action, keep on single tapping on other events, to enable multiple event selection. Also, the popup displayed at the top remains in opened state, showing the count of the number of selected events. Pressing Delete option from the popup will delete all the selected events. |

| Date navigation | Clicking on the previous or next date navigation icons in the header bar allows you to navigate between dates. | Swiping the scheduler view port to the left or right will allow you to navigate between the dates on touch devices. You can prevent the swiping action by disabling allowSwiping

Copyright © 2001 -2024 Syncfusion Inc.

property. NOTE: Swiping does not work when horizontal scroller present in the Scheduler. You can also make use of the previous and next navigation icons at the header bar to navigate. |

| View navigation | Click on an event and try moving it over the Scheduler to enable drag and drop action. | The view options are available within the popup options at the top right extreme end of the header bar and you can choose the view from it. |

| Drag and drop | Click on an event and try moving it over the Scheduler to enable drag and drop action. | Tap hold the event and try moving it over the Scheduler to enable drag and drop action. |

| Event resizing | Hover the mouse across the extremities or edges of the Scheduler events and when the mouse pointer changes into resize handler, now click and start resizing an event to the desired time range. | Touch the event extremities and start resizing the events directly. |

| Tooltip | Hover the mouse pointer over the events or resource header and the tooltip will be displayed. | Tap holding the events will open the tooltip on events. |

| Open editor window | Double click on cells or events to open the editor window. | Double click on cells or events to open the editor window. Single tap on cells, which displays a + icon on the cell. Now, tap on it again to open the new event editor window. To open the editor on events, single tap on it and then click on the edit icon to open the editor window in Edit mode. |

| Open quick info popup | Single clicking on a cell will open a quick popup prompting for new event creation. Single clicking on an event will open a popup displaying event information along with the option to edit and delete it. | No quick info popup is available while single tapping on cells. Single tapping on events, opens the popup showing event information. |

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler exampleLink to the Video to knows how to present and manipulate data.

Appointments in Vue Schedule component

Appointments can be anything that are scheduled for a specific time period. It can be created on varied time range and each appointments are categorized based on this range. The Scheduler events can be categorized as,

- Normal events
- Spanned events
- All-day events
- Recurring events

You can watch the following video to learn more about scheduling appointments and how to customize it in the Vue Scheduler

Normal events

Represents an appointment that is created for any specific time interval within a day.

Creating a normal event

The following example depicts how to define a normal event on the Scheduler, with event data being loaded from simple JSON data.

APP.VUE

```
<template>
```

```
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate' :views =
'views' :eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import {SchedulePlugin, Day, Week, TimelineViews, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
let data = [{
    Id: 1,
    Subject: 'Paris',
    StartTime: new Date(2018, 1, 15, 10, 0),
    EndTime: new Date(2018, 1, 15, 12, 30)
}];
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      views: ['Day', 'Week', 'TimelineWeek', 'Month', 'Agenda'],
      eventSettings: {
        dataSource: data
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, TimelineViews, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Spanned events

Represents an appointment that is created for more than 24 hours, and usually displayed on the all-day row. Also, represents another type of appointment that is created for more than one day but less than 24 hours, and usually displayed appropriately on both the days.

For example, if an appointment is created for two days say from November 25, 2018 – 11.00 PM to November 26, 2018 2.00 AM but less than 24 hours time interval, then the appointment is split into two partitions and will be displayed on both the days.

All-day events

Represents an appointment that is created for an entire day such as holiday events. It is usually displayed separately in an all-day row, a separate row for all-day appointments below the date header section. In Timeline views, the all-day appointments displays in the working space area, and no separate all-day row is present in that view.

To change normal appointment into all-day event, set isAllDay field to true.

Hide all-day row events

You can make use of the CSS customization to prevent the display of all-day row appointments on the Scheduler UI.

`html

<style>

.e-schedule .e-date-header-wrap .e-schedule-table thead {

display: none;

}

</style>

`

Expand all day appointments view on initial load

When you have larger number of appointments in all-day view, you can show all all-day events using dataBound event on at initial load. So, user don't have to click the toggle to expand all-day events.

APP.VUE

```
<template>
  <div id="app">
    <div id="container">
      <ejs-schedule ref='scheduleObj' :height="height"
:selectedDate="selectedDate" :views="views"
        :eventSettings="eventSettings" :dataBound=this.dataBound></ejs-
schedule>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import {
  ScheduleComponent,
  Day,
  Week,
  Month,
```

```
  Agenda,
  Inject,
  Resize,
  DragAndDrop,
  SchedulePlugin,
} from "@syncfusion/ej2-vue-schedule";
var initialLoad = true;
let data = [
  {
    EndTime: new Date(2022, 3, 30, 0, 0),
    Id: "2",
    IsAllDay: true,
    StartTime: new Date(2022, 3, 29, 0, 0),
    Subject: "Plumbing Checklist | Jaimungal | 3671 :: Pool",
  },
  {
    EndTime: new Date(2022, 3, 30, 0, 0),
    Id: "4",
    IsAllDay: true,
    StartTime: new Date(2022, 3, 28, 0, 0),
    Subject: "Underground Plumbing | Jaimungal | 3671 :: Pool",
  },
  {
    EndTime: new Date(2022, 3, 30, 12, 30),
    Id: "7",
    IsAllDay: true,
    StartTime: new Date(2022, 3, 24, 0, 0),
    Subject: "Steel/ Checklist | VP Highland Model | 3719 :: Pool",
  },
  {
    EndTime: new Date(2022, 3, 30, 0, 0),
    Id: "9",
    IsAllDay: true,
    StartTime: new Date(2022, 3, 29, 0, 0),
    Subject:
      "Tile Selections/ Pavers/ Finish | VP Highland Model | 3719 :: Pool",
  },
  {
    EndTime: new Date(2022, 3, 30, 0, 0),
    Id: "10",
    IsAllDay: true,
    StartTime: new Date(2022, 3, 26, 0, 0),
    Subject: "Layout/ Form Rebar Shell | VP Highland Model | 3719 :: Pool",
  },
  {
    EndTime: new Date(2022, 3, 30, 0, 0),
    Id: "10",
    IsAllDay: true,
    StartTime: new Date(2022, 3, 26, 0, 0),
    Subject: " VP Highland Model | 3719 :: Pool",
  },
];
Vue.use(SchedulePlugin);
export default {
  data() {
    return {
      height: "550px",
```

```
      views: ["Day", "Week", "Month", "Agenda"],
      eventSettings: {
        dataSource: data,
        spannedEventPlacement: "TimeSlot",
      },
      selectedDate: new Date(2022, 3, 26),
    };
  },
  provide: {
    schedule: [Day, Week, Month, Agenda, DragAndDrop],
  },
  methods: {
    dataBound(args) {
      let scheduleObj = this.$refs.scheduleObj.ej2Instances;
      debugger;
      if (initialLoad) {
        setTimeout(() => {
          console.log(document.querySelector('.e-all-day-appointment-
section'));
          document.querySelector('.e-all-day-appointment-section').click();
        }, 0)
        initialLoad = false;
      }
    }
  }
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs2" %}

Customize the rendering of the spanned events

By default, Scheduler will renders the spanned events (appointment with more than 24 hours duration) in the all-day row by setting `AllDayRow` will the default type renders to the `spannedEventPlacement` option within the eventSettings property. Now we can customize rendering of the that events inside the work cells itself by modifying the `spannedEventPlacement` option as `TimeSlot`. In this following example, shows how to render the spanned appointments inside the work cells as follows.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <ejs-schedule :height='height' :selectedDate='selectedDate' :views =
'views' :eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import {SchedulePlugin, Day, Week, Month, Agenda } from '@syncfusion/ej2-
vue-schedule';
let data = [{
    Id: 1,
    Subject: 'Paris',
    StartTime: new Date(2018, 1, 15, 10, 0),
    EndTime: new Date(2018, 1, 17, 12, 30),
    IsAllDay: false
},{
    Id: 2,
    Subject: 'London',
    StartTime: new Date(2018, 1, 16, 12, 0),
    EndTime: new Date(2018, 1, 18, 13, 0),
    IsAllDay: false
}];
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      views: ['Day', 'Week', 'Month', 'Agenda'],
      eventSettings: {
        dataSource: data,
        spannedEventPlacement: 'TimeSlot'
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/event-cs3" %}

Recurring events

Represents an appointment that is created for a certain time interval and occurring repeatedly on a daily, weekly, monthly or yearly basis at the same time interval based on the provided recurrence rule. Usually, the recurring events are indicated by a repeat marker added at the bottom-right position.

Creating a recurring event

The following example depicts how to create a recurring event on Scheduler with the specific recurrence rule. In the following example, an event is made to repeat on daily mode and ends after 5 occurrences.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate' :views =
'views' :eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, TimelineMonth, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
let data = [{
    Id: 1,
    Subject: 'Paris',
    StartTime: new Date(2018, 1, 15, 10, 0),
    EndTime: new Date(2018, 1, 15, 12, 30),
    IsAllDay: false,
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=5'
}];
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      views: ['Day', 'Week', 'TimelineMonth', 'Month', 'Agenda'],
      eventSettings: {
        dataSource: data
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, TimelineMonth, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs4" %}

Adding exceptions

A few instance of the recurrence series can be excluded on specific dates, by adding those exceptional dates to the recurrenceException field. These date values should be given in the ISO date time format with no hyphens(-) separating the date elements.

For example, 22nd February 2018 can be represented as 20180222. Also, the time part being represented in UTC format needs to add "Z" after the time portion with no space. "07:30:00 UTC" is therefore represented as "073000Z".

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
let data = [{
    Id: 1,
    Subject: 'Paris',
    StartTime: new Date(2018, 0, 28, 10, 0),
    EndTime: new Date(2018, 0, 28, 12, 30),
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=8',
    RecurrenceException:'20180129T043000Z,20180131T043000Z,20180202T043000Z'
}];
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: {
        dataSource: data
      },
      selectedDate: new Date(2018, 0, 28),
```

```
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs5" %}

Editing an occurrence from a series

To dynamically edit a particular occurrence from an event series and display it on the initial load of Scheduler, the edited occurrence needs to be added as a new event to the dataSource collection, with an additional `recurrenceID` field defined to it. The `recurrenceID` field of edited occurrence usually maps the ID value of the parent event.

In this example, a recurring instance that displays on the date 30th Jan 2018 is edited with different timings. Therefore, this particular date is excluded from the parent recurring event that repeats from 28th January 2018 to 4th February 2018. This can be done by adding the `recurrenceException` field with the excluded date value on the parent event. Also, the edited occurrence event which is created as a new event should carry the `recurrenceID` field pointing to the parent event's `Id` value.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
let data = [{
    Id: 1,
```

```
    Subject: 'Scrum Meeting',
    StartTime: new Date(2018, 0, 28, 10, 0),
    EndTime: new Date(2018, 0, 28, 12, 30),
    IsAllDay: false,
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=8',
    RecurrenceException:'20180130T043000Z'
},
{
    Id: 2,
    Subject: 'Scrum Meeting',
    StartTime: new Date(2018, 0, 30, 9, 0),
    EndTime: new Date(2018, 0, 30, 10, 30),
    Description: 'Meeting time changed based on team activities.',
    RecurrenceID: 1
}];
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: {
        dataSource: data
      },
      selectedDate: new Date(2018, 0, 28),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs6" %}

Edit only the current and following events

To edit only the current and following events enable the property editFollowingEvents within eventSettings property. The edited occurrence needs to be added as a new event to the dataSource

collection, with an additional followingID field defined to it. The followingID field of edited occurrence usually maps the ID value of the immediate parent event.

In this example, a recurring instance that displays on the date 30th Jan 2018 and its following dates are edited with different subject. Therefore, this particular date and its following dates are excluded from the parent recurring event that repeats from 28th January 2018 to 4th February 2018. This can be done by updating the recurrenceRule field with the until date value on the parent event. Also, the edited events which is created as a new event should carry the followingID field pointing to the immediate parent event's Id value.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
let data = [{
    Id: 1,
    Subject: 'Scrum Meeting',
    StartTime: new Date(2018, 0, 28, 10, 0),
    EndTime: new Date(2018, 0, 28, 12, 30),
    IsAllDay: false,
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;UNTIL=20180129T043000Z;',
},
{
    Id: 2,
    Subject: 'Scrum Meeting - Following Edited',
    StartTime: new Date(2018, 0, 30, 10, 0),
    EndTime: new Date(2018, 0, 30, 12, 30),
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;UNTIL=20180204T043000Z;',
    FollowingID: 1
}];
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: {
        dataSource: data,
        editFollowingEvents: true
      },
      selectedDate: new Date(2018, 0, 28),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs7" %}

Recurrence options and rules

Events can be repeated on a daily, weekly, monthly or yearly basis based on the recurrence rule which accepts the string value. The following details should be assigned to the `recurrenceRule` property to generate the recurring instances.

- Repeat type - daily/weekly/monthly/yearly.
- How many times it needs to be repeated?
- The interval duration.
- The time period to render the appointment, etc.

There are four repeat types available namely,

- **Daily** - Creates the recurring instances on daily basis.
- **Weekly** - Creates the recurring instances on weekly basis for the selected days.
- **Monthly** - Creates the recurring instances on monthly basis for the selected months and other provided recurrence criteria.
- **Yearly** - Creates the recurring instances on yearly basis.

Recurrence properties

The properties based on which the recurrence appointments are created with its respective time period are depicted in the following table. Also, the valid rule string can be referred from iCalendar specifications.

Refer iCalendar specifications for valid recurrence rule string.

Property	Purpose	Example
FREQ	Maintains the repeat type (Daily, Weekly, Monthly, Yearly) value of the appointment.	FREQ=DAILY;INTERVAL=1

Copyright © 2001 -2024 Syncfusion Inc.

| INTERVAL | Maintains the interval value of the appointments. When you create the daily appointment at an interval of 2, the appointments are rendered on the days Monday, Wednesday and Friday (Creates an appointment on all days by leaving the interval of one day gap). | FREQ=DAILY;INTERVAL=2|

| COUNT | It holds the appointment's count value. When the COUNT value is 10, then 10 instances of appointments are created in the recurrence series. | FREQ=DAILY;INTERVAL=1;COUNT=10|

| UNTIL | This property holds the end date value (in ISO format) denoting when the recurrence actually ends. | FREQ=DAILY;INTERVAL=1;UNTIL=20180530T041343Z;|

| BYDAY | It holds the day value(s), representing on which the appointments actually renders. Create the weekly appointment, and select the day(s) from the day options (Monday/Tuesday/Wednesday/Thursday/Friday/Saturday/Sunday). When Monday is selected, the first two letters of the selected day "MO" is saved in the BYDAY property. When multiple days are selected, the values are separated by commas. | FREQ=WEEKLY;INTERVAL=1;BYDAY=MO,WE;COUNT=10|

| BYMONTHDAY | This property is used to store the date value of the Month, while creating the Month recurrence appointment. When you create a Monthly recurrence appointment for every 3rd day of the month, then BYMONTHDAY holds the value 3 and creates the appointment on 3rd day of every month. | FREQ=MONTHLY;BYMONTHDAY=3;INTERVAL=1;COUNT=10|

| BYMONTH | This property is used to store the index value of the selected Month while creating the yearly appointments. When you create the yearly appointment on June month, the index value of June month 6 will get stored in the BYMONTH field. The appointment is created on every 6th month of a year. | FREQ=YEARLY;BYMONTHDAY=16;BYMONTH=6;INTERVAL=1;COUNT=10|

| BYSETPOS | This property is used to store the index value of the week. When you create the monthly appointment in second week of a month, the index value of the second week (2) is stored in BYSETPOS. | FREQ=MONTHLY;BYDAY=MO;BYSETPOS=2;COUNT=10|

The default recurrence related validation has been included for recurrence appointments similar to the one available in Outlook. The validation usually occurs during the recurrence appointment creation, editing, drag and drop or resizing of the recurrence appointments and also if any single occurrence changes.

Daily Frequency

| Description | Example |

|---|---|

| Daily recurring event that never ends | FREQ=DAILY;INTERVAL=1 |

| Daily recurring event that ends after 5 occurrences | FREQ=DAILY;INTERVAL=1;COUNT=5 |

| Daily recurring event that ends exactly on 12/12/2018 | FREQ=DAILY;INTERVAL=1;UNTIL=20181212T041343Z |

| Daily event that recurs on alternative days and repeats for 10 occurrences | FREQ=DAILY;INTERVAL=2;COUNT=10 |

Weekly Frequency

| Description | Example |

|---|---|

| Weekly recurring event that repeats on every Monday, Wednesday and Friday and never ends | FREQ=WEEKLY;INTERVAL=1;BYDAY=MO,WE,FR |

Copyright © 2001 -2024 Syncfusion Inc.

| Repeats every week Thursday and ends after 10 occurrences | FREQ=WEEKLY;INTERVAL=1;BYDAY=TH; COUNT=10 |

| Repeats every week Monday and ends on 12/12/2018 | FREQ=WEEKLY;INTERVAL=1;BYDAY=MO; UNTIL=20181212T041343Z |

| Repeats on Monday, Wednesday and Friday of alternative weeks and ends after 10 occurrences | FREQ=WEEKLY;INTERVAL=2;BYDAY=MO,WE,FR;COUNT=10 |

Monthly Frequency

| Description | Example |

|---|---|

| Monthly recurring event that repeats on every 15th day of a month and never ends | FREQ=MONTHLY; BYMONTHDAY=15;INTERVAL=1 |

| Monthly recurring event that repeats on every 16th day of a month and ends after 10 occurrences | FREQ=MONTHLY;BYMONTHDAY=16;INTERVAL=1;COUNT=10 |

| Repeats every 17th day of a month and ends on 12/12/2018 | FREQ=MONTHLY;BYMONTHDAY=17; INTERVAL=1;UNTIL=20181212T041343Z |

| Repeats every 2nd Friday of a month and never ends | FREQ=MONTHLY;BYDAY=FR;BYSETPOS=2; INTERVAL=1 |

| Repeats every 4th Wednesday of a month and ends after 10 occurrences | FREQ=MONTHLY;BYDAY=WE; BYSETPOS=4;INTERVAL=1;COUNT=10 |

| Repeats every 4th Friday of a month and ends on 12/12/2018 | FREQ=MONTHLY;BYDAY=FR;BYSETPOS=4; INTERVAL=1;UNTIL=20181212T041343Z; |

Yearly Frequency

| Description | Example |

|---|---|

| Yearly event that repeats on every 15th day of December month and never ends | FREQ=YEARLY; BYMONTHDAY=15;BYMONTH=12;INTERVAL=1 |

| Event that repeats on every 10th day of December month and ends after 10 occurrences | FREQ=YEARLY; BYMONTHDAY=10;BYMONTH=12;INTERVAL=1;COUNT=10 |

| Repeats on every 12th day of December month and ends on 12/12/2025 | FREQ=YEARLY;BYMONTHDAY=12; BYMONTH=12;INTERVAL=1;UNTIL=20251212T041343Z |

| Repeats on every 3rd Friday of December month and never ends | FREQ=YEARLY;BYDAY=FR;BYMONTH=12; BYSETPOS=3;INTERVAL=1 |

| Repeats on every 3rd Tuesday of December month and ends after 10 occurrences | FREQ=YEARLY; BYDAY=TU;BYMONTH=12;BYSETPOS=3;INTERVAL=1;COUNT=10 |

| Repeats on every 4th Wednesday of December month and ends on 12/12/2028 | FREQ=YEARLY;BYDAY=WE; BYMONTH=12;BYSETPOS=4;INTERVAL=1;UNTIL=20281212T041343Z |

Copyright © 2001 -2024 Syncfusion Inc.

Recurrence Validation

The built-in validation support has been added by default for recurring appointments during its creation, edit, drag and drop or resize action. The following are the possible validation alerts that displays on Scheduler while creating or editing the recurring events.

| Validation messages | Description |

|-------|---------|

| The recurrence pattern is not valid. | This alert will raise, when the selected recurrence rule value is not a valid one. For example, when you try to select the end date value (using `Until` option) for a recurring event, which occurs before the start date, an alert will popup out saying that the chosen pattern is invalid. |

| The changes made to specific instances of this series will be cancelled and those events will match the series again. | This alert will raise, when you try to edit the whole series, whose occurrence might have been already edited. For example, If there are five occurrences and one of the occurrence is already edited. Now, when you try to edit the entire series, you will get this validation alert. |

| The duration of the event must be shorter than how frequently it occurs. Shorten the duration, or change the recurrence pattern in the recurrence event editor. | This validation will occur, if the event duration is longer than the selected frequency. For example, if you create a recurring appointment with two days duration in `Daily` frequency with no intervals set to it, you may get this alert. |

| Some months have fewer than the selected date. For these months, the occurrence will fall on the last date of the month. | When you try to create a recurring appointment on 31st of every month, where few months won't have 31 days and in this scenario, you will get this alert. |

| Two occurrences of the same event cannot occur on the same day. | This validation will occur, when you try to edit or move any single occurrence to some other date, where another occurrence of the same event is already present. |

Event fields

The Scheduler dataSource usually holds the event instances, where each of the instance includes a collection of appropriate fields. It is mandatory to map these fields with the equivalent fields of database, when remote data is bound to it. When the local JSON data is bound, then the field names defined within the instances needs to be mapped with the scheduler event fields correctly.

To create an event on Scheduler, it is enough to define the `startTime` and `endTime`. Also `id` field becomes mandatory to process CRUD actions on appropriate events.

Built-in fields

The built-in fields available on Scheduler event object are as follows.

| Field name | Description |

|------------------|---------|

| id | The `id` field needs to be defined as mandatory and this field usually assigns a unique ID value to each of the events.|

| subject | The `subject` field is optional, and usually assigns the summary text to each of the events.|

| startTime | The `startTime` field defines the start time of an event and it is mandatory to provide it for any of the valid event objects.|

Copyright © 2001 -2024 Syncfusion Inc.

| endTime | The endTime field defines the end time of an event and it is mandatory to provide the end time for any of the valid event objects.|

| startTimezone | It maps the startTimezone field from the dataSource and usually accepts the valid IANA timezone names. It is assumed that the value provided for this field is taken into consideration while processing the startTime field. When this field is not mapped with any timezone names, then the events will be processed based on the timezone assigned to the Scheduler.|

| endTimezone | It maps the endTimezone field from the dataSource and usually accepts the valid IANA timezone names. It is assumed that the value provided for this field is taken into consideration while processing the endTime field. When this field is not mapped with any timezone names, then the events will be processed based on the timezone assigned to the Scheduler.|

| location | It maps the location field from the dataSource and the location text value will be displayed over the events.|

| description | It maps the description field from the dataSource and denotes the event description which is optional.|

| isAllDay | The isAllDay field is mapped from the dataSource and is used to denote whether an event is created for an entire day or for specific time alone. Usually, an event with isAllDay field set to true will be considered as an all-day event. |

| recurrenceID | It maps the recurrenceID field from dataSource and usually holds the ID value of the parent recurrence event. This field is applicable only for the edited occurrence events.|

| recurrenceRule | It maps the recurrenceRule field from dataSource and holds the recurrence rule value in a string format. Also, it uniquely identifies whether the event belongs to a recurring type or normal ones. |

| recurrenceException | It maps the recurrenceException field from dataSource and is used to hold the collection of exception dates, on which the recurring occurrences needs to be excluded. The recurrenceException should be specified in UTC format. |

| isReadonly | It maps the isReadonly field from dataSource. It is mainly used to make specific appointments as readonly when set to true. |

| isBlock | It maps the isBlock field from dataSource. It is used to block the particular time ranges in the Scheduler and prevents the event creation on those time slots. |

Binding different field names

When the fields of event instances has the default mapping name, it is not mandatory to map them manually. If a Scheduler's dataSource holds the events collection with different field names, then it is necessary to map them with its equivalent field name within the eventSettings property.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
let data = [{
    TravelId: 2,
    TravelSummary: 'Paris',
    DepartureTime: new Date(2018, 1, 15, 10, 0),
    ArrivalTime: new Date(2018, 1, 15, 12, 30),
    FullDay: false,
    Source: 'London',
    Comments: 'Summer vacation planned for outstation.',
    Origin: 'Asia/Yekaterinburg',
    Destination: 'Asia/Yekaterinburg'
}];
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: {
        dataSource: data,
        fields: {
            id: 'TravelId',
            subject: { name: 'TravelSummary' },
            isAllDay: { name: 'FullDay' },
            location: { name: 'Source' },
            description: { name: 'Comments' },
            startTime: { name: 'DepartureTime' },
            endTime: { name: 'ArrivalTime' },
            startTimezone: { name: 'Origin' },
            endTimezone: { name: 'Destination' }
        }
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs8" %}

The mapper field id is of string type and has no additional validation options, whereas all other fields are of Object type and has additional options.

Event field settings

Each field of the Scheduler events are provided with additional settings such as options to set default value, to map with appropriate data source fields, to validate every event fields and to provide label values for those fields in the event window.

| Options | Description |

|---|---|

| default | Accepts the default value to the applicable fields (Subject, Location and Description), when no values are provided to them from dataSource. |

| name | Accepts the field name to be mapped from the dataSource fields. |

| title | Accepts the label values to be displayed for the fields of event editor. |

| validation | Defines the validation rules to be applied on the event fields within the event editor. |

In following example, the Subject field in event editor will display its appropriate label as **Summary**. When no subject value is provided while saving an event, then the appointment will be saved with the default subject value as **Add Summary**.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
let data = [{
    TravelId: 2,
    TravelSummary: 'Paris',
    DepartureTime: new Date(2018, 1, 15, 10, 0),
    ArrivalTime: new Date(2018, 1, 15, 12, 30),
    Source: 'London',
    Comments: 'Summer vacation planned for outstation.'
}];
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      height: '550px',
      eventSettings: {
        dataSource: data,
        fields: {
            id: 'TravelId',
            subject: { name: 'TravelSummary', title: 'Summary', default:
'Add Summary' },
            location: { name: 'Source', default: 'USA' },
            description: { name: 'Comments' },
            startTime: { name: 'DepartureTime' },
            endTime: { name: 'ArrivalTime' }
        }
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs9" %}

Adding Custom fields

Apart from the default Scheduler fields, the user can include 'n' number of custom fields for appointments. The following code example shows how to include two custom fields namely **Status** and **Priority** within event collection. It is not necessary to bind the custom fields within the eventSettings. However, those additional fields can be accessed easily, for internal processing as well as from application end.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
```

```
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
let data = [{
    Id: 2,
    Subject: 'Meeting',
    StartTime: new Date(2018, 1, 15, 10, 0),
    EndTime: new Date(2018, 1, 15, 12, 30),
    IsAllDay: false,
    Status: 'Completed',
    Priority: 'High'
}];
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: {
        dataSource: data,
        fields: {
            id: 'Id',
            subject: { name: 'Subject' },
            isAllDay: { name: 'IsAllDay' },
            startTime: { name: 'StartTime' },
            endTime: { name: 'EndTime' }
        }
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/event-cs10" %}

Customize the order of the overlapping events

By default, the scheduler will render the overlapping events based on the start and end time. Now we can customize the order of the overlapping events based on the custom fields by using the sortComparer property grouped under the eventSettings property. The following code example shows how to sort the appointments based on the custom field as follows.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda,
SortComparerFunction } from '@syncfusion/ej2-vue-schedule';
let data = [{
    Id: 1,
    Subject: 'Rank 1',
    StartTime: new Date(2017, 9, 29, 10, 0),
    EndTime: new Date(2017, 9, 29, 11, 30),
    IsAllDay: false,
    RankId: '1'
}, {
    Id: 2,
    Subject: 'Rank 3',
    StartTime: new Date(2017, 9, 29, 10, 30),
    EndTime: new Date(2017, 9, 29, 12, 30),
    IsAllDay: false,
    RankId: '3'
}, {
    Id: 3,
    Subject: 'Rank 6',
    StartTime: new Date(2017, 9, 29, 7, 0),
    EndTime: new Date(2017, 9, 29, 14, 30),
    IsAllDay: false,
    RankId: '6'
}, {
    Id: 4,
    Subject: 'Rank 9',
    StartTime: new Date(2017, 9, 29, 11, 0),
    EndTime: new Date(2017, 9, 29, 15, 30),
    IsAllDay: false,
    RankId: '9'
}];
let comparerFun = (args) => {
  args.sort((event1, event2) => event1.RankId.localeCompare(event2.RankId,
undefined, { numeric: true }));
  return args;
}
Vue.use(SchedulePlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data (){
    return {
      height: '550px',
      eventSettings: {
        dataSource: data,
        sortComparer: comparerFun
      },
      selectedDate: new Date(2017, 9, 29),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs11" %}

Drag and drop appointments

Appointments can be rescheduled to any time by dragging and dropping them onto the desired location. To work with drag and drop functionality, it is necessary to inject the module DragAndDrop and make sure that allowDragAndDrop is set to true on Scheduler. In mobile mode, you can drag and drop the events by tap holding an event and dropping them on to the desired location.

By default, drag and drop action is applicable on all Scheduler views, except Agenda, Month-Agenda and Year view.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :views='views'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, TimelineViews, TimelineMonth, Month,
Agenda, DragAndDrop } from '@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['Day','Week','Month','TimelineDay','TimelineMonth','Agenda'],
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, TimelineViews, TimelineMonth, Month, Agenda,
DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs12" %}

Drag and drop multiple appointments

We can drag and drop multiple appointments by enabling the allowMultiDrag property. We can select multiple appointments by holding the CTRL key. Once the events are selected, we can leave the CTRL key and start dragging the event.

We can also drag multiple events from one resource to another resource. In this case, if all the selected events are in the different resources, then all the events should be moved to the single resource that is related to the target event.

Note: Multiple events drag and drop is not supported on mobile devices.

APP.VUE

```
<template>
```

```
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:allowMultiDrag='allowMultiDrag' :eventSettings='eventSettings'></ejs-
schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, DragAndDrop }
from '@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      allowMultiDrag: true,
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs13" %}

Disable the drag action

By default, you can drag and drop the events within any of the applicable scheduler views, and to disable it, set `false` to the allowDragAndDrop property.

APP.VUE

```
<template>
  <div id='app'>
```

```
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:allowDragAndDrop='allowDragAndDrop' :eventSettings='eventSettings'></ejs-
schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, DragAndDrop }
from '@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      allowDragAndDrop: false,
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs14" %}

Preventing drag and drop on specific targets

It is possible to prevent the drag action on particular target, by passing the target to be excluded in the excludeSelectors option within dragStart event. In this example, we have prevented the drag action on all-day row.

APP.VUE

```
<template>
  <div id='app'>
```

```
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings' :dragStart='onDragStart'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, DragAndDrop }
from '@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onDragStart: function(args) {
        args.excludeSelectors = 'e-header-cells,e-header-day,e-header-
date,e-all-day-cells';
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs15" %}

Disable scrolling on drag action

By default, while dragging an appointment to the edges, either top/bottom in the vertical Scheduler or left/right in the timeline Scheduler, scrolling action takes place automatically. To prevent this scrolling, set `false` to the `scroll` value within the dragStart event arguments.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings' :dragStart='onDragStart'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, DragAndDrop,
DragEventArgs } from '@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onDragStart: function(args) {
        args.scroll = { enable: false };
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs16" %}

Copyright © 2001 -2024 Syncfusion Inc.

Controlling scroll speed while dragging an event

The speed of the scrolling action while dragging an appointment to the Scheduler edges, can be controlled within the dragStart event by setting the desired value to the scrollBy and timeDelay option whereas its default value is 30 minutes and 100ms.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings' :dragStart='onDragStart'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, DragAndDrop,
DragEventArgs } from '@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onDragStart: function(args) {
      args.scroll = { enable: true, scrollBy: 5, timeDelay: 200 };
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/event-cs17" %}

Auto navigation of date ranges on dragging an event

When an event is dragged either to the left or right extreme edges of the Scheduler and kept hold for few seconds without dropping, the auto navigation of date ranges will be enabled allowing the Scheduler to navigate from current date range to back and forth respectively. This action is set to `false` by default and to enable it, you need to set `navigation` to true within the dragStart event.

By default, the navigation delay is set to 2000ms. The navigation delay decides how long the user needs to drag and hold the appointments at the extremities. You can also set your own delay value for letting the users to navigate based on it, using the `timeDelay` within the `dragStart` event.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings' :dragStart='onDragStart'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, DragAndDrop,
DragEventArgs } from '@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onDragStart: function(args) {
      args.navigation = { enable: true, timeDelay: 4000 }
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs18" %}

Setting drag time interval

By default, while dragging an appointment, it moves at an interval of 30 minutes. To change the dragging time interval, pass the appropriate values to the interval option within the dragStart event.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings' :dragStart='onDragStart'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, DragAndDrop,
DragEventArgs } from '@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onDragStart: function(args) {
      args.interval = 10; //drag interval time is changed to 10 minutes
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs19" %}

Drag and drop items from external source

It is possible to drag and drop the unplanned items from any of the external source into the scheduler, by manually saving those dropped item as a new appointment data through Link to the Video method of Scheduler.

Visit this video to find out more about drag and drop events from an external source into the Vue Scheduler:

In this example, we have used the tree view control as an external source and the child nodes from the tree view component are dragged and dropped onto the Scheduler. Therefore, it is necessary to make use of the nodeDragStop event of the TreeView component, where we can form an event object and save it using the addEvent method.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
      <div class='content-wrapper'>
        <div class='schedule-container'>
          <div class='title-container'>
            <h1 class='title-text'>Scheduler</h1>
          </div>
          <div>
            <ejs-schedule ref='ScheduleObj' :height='height' :width='width'
:selectedDate='selectedDate' :views = 'views' :eventSettings='eventSettings'
cssClass='schedule-drag-drop' :actionBegin='onActionBegin'
:drag='onItemDrag'></ejs-schedule>
          </div>
        </div>
        <div class='treeview-container'>
          <div class='title-container'>
            <h1 class='title-text'>Waiting List</h1>
          </div>
          <div>
            <ejs-treeview id='Tree' cssClass='treeview-external-drag'
:allowDragAndDrop=true :fields='treeViewFields' :nodeDragging='onItemDrag'
:nodeDragStop='onTreeDragStop' ></ejs-treeview>
          </div>
        </div>
      </div>
    </div>
```

```
  </div>
</template>
<style>
/* csslint ignore:start */
.content-wrapper {
  display: -ms-flexbox;
  display: flex;
}
.e-device-hover {
  background-color: #e0e0e0 !important;
}
.schedule-container {
  padding-right: 10px
}
.title-container {
  padding-bottom: 10px;
}
.treeview-external-drag .e-list-text,
.e-bigger .treeview-external-drag .e-list-text {
  background: white;
  border: 0.5px solid #E1E7EC;
  height: 30px;
  line-height: 15px;
  padding: 5px;
  width: 220px;
}
.treeview-external-drag .e-list-parent,
.e-bigger .treeview-external-drag .e-list-parent {
  height: 100%;
  padding: 0 2px;
}
.treeview-external-drag .e-list-item,
.e-bigger .treeview-external-drag .e-list-item {
  height: 100%;
  padding: 0 0 5px 0;
}
.treeview-external-drag .e-fullrow,
.e-bigger .treeview-external-drag .e-fullrow {
  height: 55px;
}
.treeview-external-drag .e-list-item.e-hover>.e-fullrow,
.treeview-external-drag .e-list-item.e-active>.e-fullrow,
.treeview-external-drag .e-list-item.e-active.e-hover>.e-fullrow,
.e-bigger .treeview-external-drag .e-list-item.e-hover>.e-fullrow,
.e-bigger .treeview-external-drag .e-list-item.e-active>.e-fullrow,
.e-bigger .treeview-external-drag .e-list-item.e-active.e-hover>.e-fullrow {
  background-color: transparent;
  border-color: transparent;
  box-shadow: none !important;
}
.treeview-external-drag .e-text-content,
.e-bigger .treeview-external-drag .e-text-content {
  padding: 0;
}
.e-drag-item.e-treeview.treeview-external-drag,
.e-bigger .e-drag-item.e-treeview.treeview-external-drag {
  padding: 0 !important;
```

```
}
.title-text {
  font-size: 18px;
  margin: 0px;
  font-weight: bold;
  text-align: center;
}
@media (max-width: 550px) {
  .content-wrapper {
    display: block;
  }
  .schedule-container {
    padding-bottom: 10px
  }
  .treeview-external-drag.e-treeview,
  .e-bigger .treeview-external-drag.e-treeview {
    width: 225px;
  }
  .e-bigger .treeview-external-drag.e-treeview.e-drag-item {
    position: relative !important;
  }
}
/* csslint ignore:end */
</style>
<script>
import Vue from 'vue';
import { closest, remove, addClass } from '@syncfusion/ej2-base';
import { SchedulePlugin, Month, Resize, DragAndDrop } from '@syncfusion/ej2-
vue-schedule';
import { eventData, waitingList } from './datasource.js';
Vue.use(SchedulePlugin);
import { TreeViewPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(TreeViewPlugin);
export default {
  data (){
    return {
      height: '510px',
      width: '100%',
      views: ['Month'],
      eventSettings: {
        dataSource: eventData
      },
      selectedDate: new Date(2018, 1, 15),
      treeViewFields: { dataSource: waitingList, id: 'Id', text: 'Name' },
      draggedItemId : null
    }
  },
  methods: {
    onActionBegin: function (event) {
      if (event.requestType === 'eventCreate' && this.draggedItemId) {
        let treeObj = document.getElementById('Tree').ej2_instances[0];
        let treeViewdata = treeObj.fields.dataSource;
        let filteredPeople = treeViewdata.filter(this.dataFilter);
        treeObj.fields.dataSource = filteredPeople;
        let elements = document.querySelectorAll('.e-drag-item.treeview-
external-drag');
        for (let i = 0; i < elements.length; i++) {
```

```
            remove(elements[i]);
          }
        }
      },
      onItemDrag: function(event) {
        let scheduleObj = this.$refs.ScheduleObj.ej2Instances;
        if (scheduleObj.isAdaptive) {
          let classElement = document.querySelector('.e-device-hover');
          if (classElement) {
            classElement.classList.remove('e-device-hover');
          }
          if (event.target.classList.contains('e-work-cells')) {
            addClass([event.target], 'e-device-hover');
          }
        }
        if (document.body.style.cursor === 'not-allowed') {
          document.body.style.cursor = '';
        }
        if (event.name == 'nodeDragging') {
          let dragElementIcon = document.querySelectorAll('.e-drag-item .e-
icon-expandable');
          for (let i = 0; i < dragElementIcon.length; i++) {
            dragElementIcon[i].style.display = 'none';
          }
        }
      },
      dataFilter: function (item) {
        return item.Id !== parseInt(this.draggedItemId, 10);
      },
      onTreeDragStop: function(event) {
        let treeElement = closest(event.target, '.e-treeview');
        let classElement = document.querySelector('.e-device-hover');
        if (classElement) {
          classElement.classList.remove('e-device-hover');
        }
        if (!treeElement) {
          event.cancel = true;
          let scheduleElement = closest(event.target, '.e-content-wrap');
          if (scheduleElement) {
            let treeObj = document.getElementById('Tree').ej2_instances[0];
            let treeviewData = treeObj.fields.dataSource;
            if (event.target.classList.contains('e-work-cells')) {
              let filteredData = treeviewData.filter(function (item) { return
item.Id === parseInt(event.draggedNodeData.id, 10); });
              let scheduleObj = this.$refs.ScheduleObj.ej2Instances;
              let cellData = scheduleObj.getCellDetails(event.target);
              let eventData = {
                Subject: filteredData[0].Name,
                StartTime: cellData.startTime,
                EndTime: cellData.endTime,
                IsAllDay: cellData.isAllDay,
                Description: filteredData[0].Description
              };
              this.draggedItemId = event.draggedNodeData.id;
              scheduleObj.addEvent(eventData);
            }
          }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      }
    }
  },
  provide: {
    schedule: [Month, Resize, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/external-drag-cs1" %}

Opening the editor window on drag stop

There are scenarios where you want to open the editor filled with data on newly dropped location and may need to proceed to save it, only when Save button is clicked on the editor. On clicking the cancel button should revert these changes. This can be achieved using the dragStop event of Scheduler.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule ref='scheduleObj' :height='height'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:dragStop='onDragStop'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, DragAndDrop }
from '@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
```

```
    }
  },
  methods: {
    onDragStop: function(args) {
        args.cancel = true; //cancels the drop action
        this.$refs.scheduleObj.ej2Instances.openEditor(args.data, 'Save');
//open the event window with updated start and end time
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs20" %}

Inline Appointment

In Scheduler, another easier way for adding or editing the appointment's subject alone can be achieved by using the inline Add/Edit support. It allows the user to add and edit the appointments inline. To get familiar with the inline Add mode, single click on any of the Scheduler cells or press enter key on the selected cells.

When the inline adding mode is ON, a text box will get created within the clicked Scheduler cells with a blinking cursor in it, requiring the user to enter the subject of an appointment. Once the subject is entered, the appointment will be saved on pressing the enter key.

To enable the inline edit mode, single click on any of the existing appointment's subject, so that the user can edit the subject of that appointment. The edited subject of that appointment will be updated on pressing the enter key.

The inline option can be enabled/disabled on the Scheduler by using the allowInline API, whereas its default value is set to false.

While using the allowInline the showQuickInfo will be turned off. The quickPopup will not show on clicking the work cell or clicking the appointment when the allowInline property is set to true.

Copyright © 2001 -2024 Syncfusion Inc.

In work cells, select multiple cells using keyboard, and then press enter key. The appointment wrapper will be created, and focus will be on the subject field. Also, consider the overlapping scenarios when creating an inline event.

Normal Event

While editing appointments, single-click the appointment subject, the editable option will be enabled in UI and the cursor will focus at the end of the text. Inline editing will be considered for all possible views.

Recurrence Event

While editing the occurrence from the recurrence series, it is only possible to edit a single occurrence, not an entire series.

APP.VUE

```
<template>
    <div id='app'>
        <ejs-schedule height='550px' :selectedDate='selectedDate'
:eventSettings='eventSettings' :allowInline='allowInline'
:currentView='currentView'>
        </ejs-schedule>
    </div>
</template>
<script>
  import Vue from 'vue';
  import { scheduleData } from './datasource.js';
  import { SchedulePlugin, Day, WorkWeek, Agenda, Month, Week } from
'@syncfusion/ej2-vue-schedule';
  Vue.use(SchedulePlugin);
  export default {
    data () {
      return {
        selectedDate: new Date(2018, 1, 17),
        allowInline: true,
        eventSettings: { dataSource: scheduleData },
        currentView: 'Month',
      }
    },
    provide: {
      schedule: [Day, WorkWeek, Agenda, Month, Week]
    }
  }
</script>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs21" %}

Appointment Resizing

Another way of rescheduling an appointment can be done by resizing it through either of its handlers. To work with resizing functionality, it is necessary to inject the module Resize and make sure that allowResizing property is set to true.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, Resize } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, Resize]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs22" %}

Disable the resize action

By default, resizing of events is allowed on all Scheduler views except Agenda and Month-Agenda view. To disable this event resizing action, set false to the allowResizing property.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:allowResizing='allowResizing' :eventSettings='eventSettings'></ejs-
schedule>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, Resize } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      allowResizing: false,
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, Resize]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs23" %}

Disable scrolling on resize action

By default, while resizing an appointment, when its handler reaches the extreme edges of the Scheduler, scrolling action will takes place along with event resizing. To prevent this scrolling action, set false to `scroll` value within the resizeStart event.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:resizeStart='onResizeStart' :eventSettings='eventSettings'></ejs-schedule>
    </div>
```

```
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, Resize } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onResizeStart: function(args) {
      args.scroll = { enable: false };
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, Resize]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs24" %}

Controlling scroll speed while resizing an event

The speed of the scrolling action while resizing an appointment to the Scheduler edges, can be controlled within the resizeStart event by setting the desired value to the scrollBy option.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
```

```
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:resizeStart='onResizeStart' :eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, Resize } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onResizeStart: function(args) {
      args.scroll = { enable: true, scrollBy: 15 };
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, Resize]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs25" %}

Setting resize time interval

By default, while resizing an appointment, it extends or shrinks at an interval of 30 minutes. To change this default resize interval, set appropriate values to interval option within the resizeStart event.

APP.VUE

```
<template>
```

```
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:resizeStart='onResizeStart' :eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, Resize } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onResizeStart: function(args) {
      args.interval = 10; //resize interval time is changed to 10 minutes
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, Resize]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs26" %}

Appointment customization

The look and feel of the Scheduler events can be customized using any one of the following ways.

- Using event template

- Using eventRendered event
- Using custom CSS class

Using template

Any kind of text, images and links can be added to customize the look of the events. The user can format and change the default appearance of the events by making use of the template option available within the Link to the Video property.

Check out the following video to learn how to customise events using templates in the Vue Scheduler:

The following code example customizes the appointment's default color and time format.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate'
        readonly='true' :eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { Internationalization } from '@syncfusion/ej2-base';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { webinarData } from './datasource.js';
Vue.use(SchedulePlugin);
var instance = new Internationalization();
var eventTemplateVue = Vue.component('eventTemplate', {
  template:`<div class='template-wrap' :style='{background:
data.SecondaryColor}'>
        <div class='subject'
:style='{background:data.PrimaryColor}'>{{data.Subject}}</div>
        <div class='time' :style='{background:data.PrimaryColor}'>Time:
{{getTimeString(data.StartTime)}} -
{{getTimeString(data.EndTime)}}</div></div>`,
  data() {
    return {
      data: {}
    };
  },
  methods: {
    getTimeString: function (value) {
      return instance.formatDate(value, { skeleton: 'hm' });
    }
  }
});
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      eventSettings: {
```

```
        dataSource: webinarData,
        template: function (e) {
          return {
            template: eventTemplateVue
          };
        }
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
  .e-schedule .e-vertical-view .e-content-wrap .e-appointment {
    border-radius: 8px;
  }
  .e-schedule .e-vertical-view .e-content-wrap .e-appointment .e-
appointment-details {
    padding: 0;
    height: 100%;
  }
  .e-schedule .template-wrap {
    height: 100%;
    white-space: normal;
  }
  .e-schedule .template-wrap .subject {
    font-weight: 600;
    font-size: 15px;
    padding: 4px 4px 4px;
    text-overflow: ellipsis;
    white-space: nowrap;
    overflow: hidden;
  }
  .e-schedule .template-wrap .time {
    height: 50px;
    font-size: 12px;
    padding: 4px 6px 4px;
    overflow: hidden;
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-template-cs1" %}

All the built-in fields that are mapped to the appropriate field properties within the eventSettings, as well as custom mapped fields from the Scheduler dataSource can be accessed within the template code.

Using eventRendered event

The eventRendered event triggers before the appointment renders on the Scheduler. Therefore, this client-side event can be utilized to customize the look of events based on any specific criteria, before rendering them on the scheduler.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :views='views' cssClass= 'custom-class'
:eventSettings='eventSettings' :eventRendered='onEventRendered'></ejs-
schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'WorkWeek', 'Month', 'Agenda'],
      eventSettings: {
        dataSource: scheduleData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods:{
    onEventRendered: function(args) {
      let categoryColor = args.data.CategoryColor;
      if (!args.element || !categoryColor) {
          return;
      }
      if (this.currentView === 'Agenda') {
          args.element.firstChild.style.borderLeftColor = categoryColor;
      } else {
          args.element.style.backgroundColor = categoryColor;
      }
    }
  },
  provide: {
```

```
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs27" %}

Using cssClass

The customization of events can also be achieved using cssClass property of the Scheduler. In the following example, the background of appointments has been changed using the cssClass.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :views='views' cssClass= 'custom-class'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'WorkWeek', 'Month', 'Agenda'],
      eventSettings: {
        dataSource: scheduleData
      },
      selectedDate: new Date(2018, 1, 15),
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
  .custom-class.e-schedule .e-vertical-view .e-all-day-appointment-wrapper
.e-appointment,
  .custom-class.e-schedule .e-vertical-view .e-day-wrapper .e-appointment,
  .custom-class.e-schedule .e-month-view .e-appointment{
    background: #006400;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs28" %}

Setting minimum height

It is possible to set minimal height for appointments on Scheduler using `eventRendered` event, when its start and end time duration is less than the default duration of a single slot.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule ref='scheduleObj' :height='height' :width='width'
:selectedDate='selectedDate' :views='views' :eventSettings='eventSettings'
:eventRendered='onEventRendered'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
vue-schedule';
Vue.use(SchedulePlugin);
var data = [{
        Id: 13,
        Subject: 'Myths of Andromeda Galaxy',
        StartTime: new Date(2018, 1, 16, 10, 30),
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        EndTime: new Date(2018, 1, 16, 10, 40)
    }, {
        Id: 14,
        Subject: 'Aliens vs Humans',
        StartTime: new Date(2018, 1, 15, 10, 0),
        EndTime: new Date(2018, 1, 15, 10, 20)
    }];
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
        dataSource: data
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods:{
    onEventRendered: function(args) {
      let scheduleObj = this.$refs.scheduleObj.ej2Instances;
      if (scheduleObj.currentView !== 'Month') {
        let cellHeight = scheduleObj.element.querySelector('.e-work-
cells').offsetHeight;
        let appHeight = (args.data.EndTime.getTime() -
args.data.StartTime.getTime()) / (60 * 1000) * (cellHeight *
scheduleObj.timeScale.slotCount) / scheduleObj.timeScale.interval;
        args.element.style.height = appHeight+'px';
      }
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs29" %}

Copyright © 2001 -2024 Syncfusion Inc.

Block Dates and Times

It is possible to block a set of dates or a particular time ranges on the Scheduler. To do so, define an appointment object within eventSettings along with the required time range to block and set the isBlock field to true. Usually, the event objects defined with isBlock field set to true will block the entire time cells lying within the appropriate time ranges specified through startTime and endTime fields.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :views='views'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, TimelineViews, Month, Agenda, Resize,
DragAndDrop } from '@syncfusion/ej2-vue-schedule';
import { blockData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'TimelineWeek', 'Month', 'Agenda'],
      eventSettings: {
        dataSource: blockData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, TimelineViews, Month, Agenda, Resize, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs30" %}

Block events can also be defined to repeat on several days as shown in the following code example.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :views='views'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, TimelineViews, Month, Agenda, Resize,
DragAndDrop } from '@syncfusion/ej2-vue-schedule';
Vue.use(SchedulePlugin);
var data = [{
    Id: 1,
    Subject: 'Explosion of Betelgeuse Star',
    StartTime: new Date(2018, 1, 15, 9, 30),
    EndTime: new Date(2018, 1, 15, 11, 0),
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=5',
    IsBlock: true
}, {
    Id: 2,
    Subject: 'Thule Air Crash Report',
    StartTime: new Date(2018, 1, 14, 12, 0),
    EndTime: new Date(2018, 1, 14, 14, 0)
}];
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'TimelineWeek', 'Month', 'Agenda'],
      eventSettings: {
        dataSource: data
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, TimelineViews, Month, Agenda, Resize, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs31" %}

Readonly

An interaction with the appointments of Scheduler can be enabled/disabled using the `readonly property. With this property enabled, you can simply navigate between the Scheduler dates, views and can be able to view the appointment details in the quick info window. Most importantly, the users are not allowed to perform any CRUD actions on Scheduler, when this property is set to true. By default, it is set as false`.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' readonly='true'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      eventSettings: {
        dataSource: scheduleData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs32" %}

Make specific events readonly

There are scenarios where you need to restrict the CRUD action on specific appointments alone based on certain conditions. In the following example, the events that has occurred on the past hours from the current date of the Scheduler are made as read-only and the CRUD actions has been prevented only on those appointments. This can be achieved by setting isReadonly field of read-only events to true.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :views='views'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek } from '@syncfusion/ej2-vue-
schedule';
import { readonlyData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['Day','Week','WorkWeek'],
      eventSettings: {
        dataSource: readonlyData
      },
      selectedDate: new Date(2018, 1, 15),
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
  provide: {
    schedule: [Day, Week, WorkWeek]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs33" %}

By default, the event editor is prevented to open on the read-only events when isReadonly field is set to true.

Restricting event creation on specific time slots

You can restrict the users to create and update more than one appointment on specific time slots. Also, you can disable the CRUD action on those time slots if it is already occupied, which can be achieved using Scheduler's public method isSlotAvailable.

Note: The **isSlotAvailable** is centered around verifying appointments within the present view's date range. Yet, it does not encompass an evaluation of availability for recurrence occurrences that fall beyond this particular date range.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule ref='scheduleObj' :height='height' :width='width'
:selectedDate='selectedDate' :views='views' :currentView='currentView'
:eventSettings='eventSettings' :actionBegin='onActionBegin'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { isNullOrUndefined } from '@syncfusion/ej2-vue-base';
import { SchedulePlugin, Day, TimelineViews, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
import { readonlyData } from './datasource.js';
Vue.use(SchedulePlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'TimelineWeek', 'WorkWeek', 'Month'],
      currentView: 'TimelineWeek',
      eventSettings: {
        dataSource: readonlyData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onActionBegin: function(args) {
      if ((args.requestType === 'eventCreate' || args.requestType ===
'eventChange') &&
      (args.data.length > 0 || !isNullOrUndefined(args.data))) {
        let eventData = args.data;
        let scheduleObj = this.$refs.scheduleObj.ej2Instances;
        let eventField= scheduleObj.eventFields;
        let startDate = (args.data.length > 0) ?
eventData[0][eventField.startTime] : eventData[eventField.startTime];
        let endDate= (args.data.length > 0) ?
eventData[0][eventField.endTime] : eventData[eventField.endTime];
        args.cancel = !scheduleObj.isSlotAvailable(startDate, endDate);
      }
    }
  },
  provide: {
    schedule: [Day, TimelineViews, WorkWeek, Month]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs34" %}

Copyright © 2001 -2024 Syncfusion Inc.

Differentiate the past time events

To differentiate the appearance of the appointments based on specific criteria such as displaying the past hour appointments with different colors on Scheduler, eventRendered event can be used which triggers before the appointment renders on the Scheduler.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule ref='scheduleObj' :height='height' :width='width'
:selectedDate='selectedDate' :views='views' cssClass= 'custom-class'
:eventSettings='eventSettings' :eventRendered='onEventRendered'></ejs-
schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
        dataSource: scheduleData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods:{
    isReadOnly: function(data) {
      return (data.EndTime <
this.$refs.scheduleObj.ej2Instances.selectedDate);
    },
    onEventRendered: function(args) {
      if (this.isReadOnly(args.data)) {
        args.element.classList.add('e-past-app');
      }
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
  .custom-class.e-schedule .e-vertical-view .e-day-wrapper .e-appointment.e-
past-app {
    background-color: #D2691E;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs35" %}

Appointments occupying entire cell

The Scheduler allows the event to occupies the full height of the cell without its header part by setting `true` for `enableMaxHeight` Property.

We can show more indicator if more than one appointment is available in a same cell by setting `true` to `enableIndicator` property whereas its default value is false.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule ref='scheduleObj' :height='height' :width='width'
:selectedDate='selectedDate' :currentView='currentView' :views='views'
:eventSettings='eventSettings'>
        </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['TimelineDay', 'TimelineMonth'],
      currentView: 'TimelineMonth',
      eventSettings: {
        dataSource: scheduleData,
        enableMaxHeight: true,
        enableIndicator: false
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [TimelineViews, TimelineMonth]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs36" %}

How to limit maximum number of events to display

In the Scheduler, the default behavior is to display concurrent events based on cell height, with each new event represented as

+n more characters. However, you may want to improve the quality of the presentation by limiting the number of concurrent events. This can be accomplished by using the maxEventsPerRow property, which is defaulted to the views property.

The maxEventsPerRow property is specific to the month, timeline month, and timeline year views, allowing you to view events visually in these rows. Below is a code example that demonstrates how to use this constraint and the events displayed in a cell have been created:

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings'>
          <e-views>
            <e-view option="Month" :maxEventsPerRow='maxEventsPerRow'></e-
view>
          </e-views>
        </ejs-schedule>
    </div>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import {SchedulePlugin, Month, Resize, DragAndDrop } from '@syncfusion/ej2-
vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '380px',
      maxEventsPerRow: 2,
      eventSettings: {
        dataSource: scheduleData
      },
      selectedDate: new Date(2023, 11, 15),
    }
  },
  provide: {
    schedule: [Month, Resize, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/max-events-per-row-cs1" %}

The property maxEventsPerRow will be applicable only when rowAutoHeight feature is disabled in the Scheduler.

Display tooltip for appointments

The tooltip shows the Scheduler appointment's information in a formatted style by making use of the tooltip related options.

Show or hide built-in tooltip

The tooltip can be displayed for appointments by setting `true` to the `enableTooltip` option within the eventSettings property.

APP.VUE

```
<template>
```

```
  <div id='app'>
    <div id='container'>
        <ejs-schedule ref='scheduleObj' :height='height' :width='width'
:selectedDate='selectedDate' :views='views'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, TimelineViews, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['TimelineDay', 'Week', 'WorkWeek', 'Month', 'Agenda'],
      eventSettings: {
        dataSource: scheduleData,
        enableTooltip: true
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [TimelineViews, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs37" %}

Customizing event tooltip using template

After enabling the default tooltip, it is possible to customize the display of needed event information on tooltip by making use of the `tooltipTemplate` option within the eventSettings.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule ref='scheduleObj' :height='height' :width='width'
:selectedDate='selectedDate' :views='views'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, TimelineViews, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
import { eventsData } from './datasource.js';
Vue.use(SchedulePlugin);
var eventTooltipTemplateVue = Vue.component('eventTooltipTemplate', {
  template:`<div class='tooltip-wrap'>
        <div class='content-area'><div
class='name'>{{data.Subject}}</></div>
        <div v-if='data.City!== null && data.City!==undefined'
class='city'>{{data.City}}</div>
        <div class='time'>From : {{data.StartTime.toLocaleString()}} </div>
        <div class='time'>To   : {{data.EndTime.toLocaleString()}}
</div></div></div>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['TimelineDay', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
        dataSource: eventsData,
        enableTooltip: true,
        tooltipTemplate: function (e) {
          return {
            template: eventTooltipTemplateVue
          };
        }
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [TimelineViews, Week, WorkWeek, Month]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/tooltip-cs1" %}

All the field names that are mapped from the Scheduler dataSource to the appropriate field properties such as subject, description, location, startTime and endTime within the eventSettings can be accessed within the template.

Appointment filtering

The appointments can be filtered by passing the predicate value to query option in eventSettings. The following code example shows how to filter and render the selected appointments alone in the Scheduler.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <table id='property' title="Filter events">
                <tbody>
                    <tr>
                        <td>
                            <ejs-checkbox ref='confirmedObj'
label='Confirmed' :checked='true' :change='onChange'></ejs-checkbox>
                        </td>
                        <td>
                            <ejs-checkbox ref='requestedObj'
label='Requested' :checked='true' :change='onChange'></ejs-checkbox>
                        </td>
                        <td>
                            <ejs-checkbox ref='newObj' label='New'
:checked='true' :change='onChange'></ejs-checkbox>
                        </td>
                    </tr>
                </tbody>
            </table>
            <ejs-schedule ref='scheduleObj' :height='height' :width='width'
:selectedDate='selectedDate'
            :eventSettings='eventSettings'
:eventRendered='onEventRendered'></ejs-schedule>
        </div>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { Query, Predicate } from '@syncfusion/ej2-data';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { CheckBoxPlugin } from '@syncfusion/ej2-vue-buttons';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
Vue.use(CheckBoxPlugin);
export default {
    data (){
        return {
            height: '525px',
            width: '100%',
            eventSettings: {
                dataSource: scheduleData
            },
            selectedDate: new Date(2018, 1, 15),
        }
    },
    methods: {
        onEventRendered: function(args) {
            switch (args.data.EventType) {
                case 'Requested':
                    args.element.style.backgroundColor = '#F57F17';
                    break;
                case 'Confirmed':
                    args.element.style.backgroundColor = '#7fa900';
                    break;
                case 'New':
                    args.element.style.backgroundColor = '#8e24aa';
                    break;
            }
        },
        onChange: function(args) {
            let predicate;
            let checkBoxes = [this.$refs.confirmedObj.ej2Instances,
this.$refs.requestedObj.ej2Instances, this.$refs.newObj.ej2Instances];
            checkBoxes.forEach((checkBoxObj) => {
                if (checkBoxObj.checked) {
                    if (predicate) {
                        predicate = predicate.or('EventType', 'equal',
checkBoxObj.label);
                    } else {
                        predicate = new Predicate('EventType', 'equal',
checkBoxObj.label);
                    }
                }
            });
            this.$refs.scheduleObj.ej2Instances.eventSettings.query = new
Query().where(predicate);
        }
    },
    provide: {
        schedule: [Day, Week, WorkWeek, Month, Agenda]
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
  #property td {
    padding: 0 5px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-filter-cs1" %}

Appointment selection

Appointment selection can be done either through mouse or keyboard actions. The selected events in UI will have a box shadow effect around to differentiate it from other appointments.

| Action | Description |

|-------|---------|

| Mouse click or Single tap on appointments | Selects single appointment. |

| Ctrl + [Mouse click] or [Single tap] on appointments | Selects multiple appointments.|

Deleting multiple appointments

With the options available to select multiple appointments, it is also possible to delete the multiple selected appointments simply by pressing the `delete` key. In case of deleting multiple selected occurrences of an event series, only those occurrences will be deleted and not the entire series.

Retrieve event details from the UI of an event

It is possible to access the information about the event fields of an appointment element displayed on the Scheduler UI. This can be achieved by passing an appointment element as argument to the public method `getEventDetails`.

In the following example, the subject of the appointment clicked has been displayed.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container' class='col-lg-12'>
        <div class='content-wrapper'>
            <div class='col-lg-9 control-section'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <ejs-schedule ref='scheduleObj' :height='height'
:width='width' :selectedDate='selectedDate' :eventSettings='eventSettings'
:eventClick='onEventClick'></ejs-schedule>
            </div>
            <div class='col-lg-3 property-section'>
                <table id='property' title='Event Trace'>
                    <tbody>
                        <tr>
                        <td>
                            <div class='eventarea' style='height:
245px;overflow: auto'>
                            <span class='EventLog' id='EventLog'
style='word-break: normal;'></span>
                            </div>
                        </td>
                        </tr>
                        <tr>
                        <td>
                            <div class='evtbtn' style='padding-bottom:
10px'>
                            <ejs-button v-
on:click.native='onClearClick'>Clear</ejs-button>
                            </div>
                        </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { ButtonPlugin} from '@syncfusion/ej2-vue-buttons';
import { eventData } from './datasource.js';
Vue.use(SchedulePlugin);
Vue.use(ButtonPlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      eventSettings: {
        dataSource: eventData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onClearClick: function (){
      document.getElementById('EventLog').innerHTML = '';
    },
    onEventClick: function (args){
```

```
      let event =
this.$refs.scheduleObj.ej2Instances.getEventDetails(args.element);
      this.appendElement(event.Subject +'<hr>');
    },
    appendElement: function(html) {
      let span = document.createElement('span');
      span.innerHTML = html;
      let log = document.getElementById('EventLog');
      log.insertBefore(span, log.firstChild);
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
  .content-wrapper {
    display: flex;
  }
  .control-section {
    padding-right: 10px;
    width:70%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/events-public-cs1" %}

Get the current view appointments

To retrieve the appointments present in the current view of the Scheduler, you can make use of the `getCurrentViewEvents` public method. In the following example, the count of current view appointment collection rendered has been traced in dataBound event.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container' class='col-lg-12'>
        <div class='content-wrapper'>
            <div class='col-lg-9 control-section'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <ejs-schedule ref='scheduleObj' :height='height'
:width='width' :selectedDate='selectedDate' :eventSettings='eventSettings'
:dataBound='onDataBound'></ejs-schedule>
            </div>
            <div class='col-lg-3 property-section'>
                <table id='property' title='Event Trace'>
                    <tbody>
                        <tr>
                        <td>
                            <div class='eventarea' style='height:
245px;overflow: auto'>
                            <span class='EventLog' id='EventLog'
style='word-break: normal;'></span>
                            </div>
                        </td>
                        </tr>
                        <tr>
                        <td>
                            <div class='evtbtn' style='padding-bottom:
10px'>
                            <ejs-button v-
on:click.native='onClearClick'>Clear</ejs-button>
                            </div>
                        </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { ButtonPlugin} from '@syncfusion/ej2-vue-buttons';
import { eventData } from './datasource.js';
Vue.use(SchedulePlugin);
Vue.use(ButtonPlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      eventSettings: {
        dataSource: eventData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onClearClick: function (){
      document.getElementById('EventLog').innerHTML = '';
    },
    onDataBound: function (args){
      let event= this.$refs.scheduleObj.ej2Instances.getCurrentViewEvents();
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      if (event.length > 0) {
        this.appendElement('Events present on current view <b>' +
event.length +'<b><hr>');
      } else {
        this.appendElement('No Events available in this view.<hr>');
      }
    },
    appendElement: function(html) {
      let span = document.createElement('span');
      span.innerHTML = html;
      let log = document.getElementById('EventLog');
      log.insertBefore(span, log.firstChild);
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
  .content-wrapper {
    display: flex;
  }
  .control-section {
    padding-right: 10px;
    width:70%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/events-public-cs2" %}

Get the entire appointment collections

The entire collection of appointments rendered on the Scheduler can be accessed using the getEvents public method. In the following example, the count of entire appointment collection rendered on the Scheduler has been traced in dataBound event.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container' class='col-lg-12'>
```

```
        <div class='content-wrapper'>
            <div class='col-lg-9 control-section'>
                <ejs-schedule ref='scheduleObj' :height='height'
:width='width' :selectedDate='selectedDate' :eventSettings='eventSettings'
:dataBound='onDataBound'></ejs-schedule>
            </div>
            <div class='col-lg-3 property-section'>
                <table id='property' title='Event Trace'>
                    <tbody>
                        <tr>
                        <td>
                            <div class='eventarea' style='height:
245px;overflow: auto'>
                            <span class='EventLog' id='EventLog'
style='word-break: normal;'></span>
                            </div>
                        </td>
                        </tr>
                        <tr>
                        <td>
                            <div class='evtbtn' style='padding-bottom:
10px'>
                            <ejs-button v-
on:click.native='onClearClick'>Clear</ejs-button>
                            </div>
                        </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { ButtonPlugin} from '@syncfusion/ej2-vue-buttons';
import { eventData } from './datasource.js';
Vue.use(SchedulePlugin);
Vue.use(ButtonPlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      eventSettings: {
        dataSource: eventData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onClearClick: function (){
      document.getElementById('EventLog').innerHTML = '';
    },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    onDataBound: function (args){
      let event = this.$refs.scheduleObj.ej2Instances.getEvents();
      this.appendElement('Events present on scheduler <b>' + event.length
+'<b><hr>');
    },
    appendElement: function(html) {
      let span = document.createElement('span');
      span.innerHTML = html;
      let log = document.getElementById('EventLog');
      log.insertBefore(span, log.firstChild);
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
  .content-wrapper {
    display: flex;
  }
  .control-section {
    padding-right: 10px;
    width:70%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/events-public-cs3" %}

Refresh appointments

If your requirement is to simply refresh the appointments instead of refreshing the entire Scheduler elements from your application end, make use of the refreshEvents public method.

`

scheduleObj.refreshEvents();

`

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Copyright © 2001 -2024 Syncfusion Inc.

Data binding in Vue Schedule component

The Scheduler uses DataManager, which supports both RESTful data service binding and JavaScript object array binding. The dataSource property of Scheduler can be assigned either with the instance of DataManager or JavaScript object array collection, as it supports the following two kind of data binding methods:

- Local data
- Remote data

Binding local data

To bind local JSON data to the Scheduler, you can simply assign a JavaScript object array to the dataSource option of the scheduler within the eventSettings property. The JSON object dataSource can also be provided as an instance of DataManager and assigned to the Scheduler dataSource property.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedData'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
Vue.use(SchedulePlugin);
let scheduleData = [{
    Id: 1,
    Subject: 'Explosion of Betelgeuse Star',
    StartTime: new Date(2018, 1, 15, 9, 30),
    EndTime: new Date(2018, 1, 15, 11, 0)
}, {
    Id: 2,
    Subject: 'Thule Air Crash Report',
    StartTime: new Date(2018, 1, 12, 12, 0),
    EndTime: new Date(2018, 1, 12, 14, 0)
}, {
    Id: 3,
    Subject: 'Blue Moon Eclipse',
    StartTime: new Date(2018, 1, 13, 9, 30),
    EndTime: new Date(2018, 1, 13, 11, 0)
}, {
    Id: 4,
    Subject: 'Meteor Showers in 2018',
    StartTime: new Date(2018, 1, 14, 13, 0),
    EndTime: new Date(2018, 1, 14, 14, 30)
}];
export default {
  data (){
    return {
      height: '550px',
```

```
      eventSettings: {
          dataSource: scheduleData
      },
      selectedData: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/data-bind-cs1" %}

By default, DataManager uses JsonAdaptor for binding local data.

You can also bind different field names to the default event fields as well as include additional custom fields to the event object collection which can be referred here.

Binding remote data

Any kind of remote data services can be bound to the Scheduler. To do so, create an instance of DataManager and provide the service URL to the url option of DataManager and then assign it to the dataSource property within eventSettings.

Using ODataV4Adaptor

ODataV4 is a standardized protocol for creating and consuming data. Refer to the following code example to retrieve the data from ODataV4 service using the DataManager. To connect with ODataV4 service end points, it is necessary to make use of ODataV4Adaptor within DataManager.

APP.VUE

```
<template>
    <div id='app'>
      <div id='container'>
        <ejs-schedule height='550px' :selectedDate='selectedDate'
:eventSettings='eventSettings'
        :readOnly='readOnly'></ejs-schedule>
      </div>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    import Vue from 'vue';
    import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-
data';
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, View } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    let dataManager = new DataManager({
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/',
        adaptor: new ODataV4Adaptor(),
        crossDomain: true
    });
    export default {
        data() {
            return {
                selectedDate: new Date(1996, 6, 9),
                readOnly: true,
                eventSettings: { dataSource: dataManager,
                    fields: {
                        id: 'Id',
                        subject: { name: 'ShipName' },
                        location: { name: 'ShipCountry' },
                        description: { name: 'ShipAddress' },
                        startTime: { name: 'OrderDate' },
                        endTime: { name: 'RequiredDate' },
                        recurrenceRule: { name: 'ShipRegion' }
                    }
                }
            };
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda]
        }
    }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/data-bind-cs2" %}

Copyright © 2001 -2024 Syncfusion Inc.

Filter events using the in-built query

To enable server-side filtering operations based on predetermined conditions, the includeFiltersInQuery API can be set to true, this allows the filter query to be constructed using the start date, end date, and recurrence rule which in turn enables the request to be filtered accordingly.

This method greatly improves the component's performance by reducing the data that needs to be transferred to the client side. As a result, the component's efficiency and responsiveness are significantly enhanced, resulting in a better user experience. However, it is important to consider the possibility of longer query strings, which may cause issues with the maximum URL length or server limitations on query string length.

APP.VUE

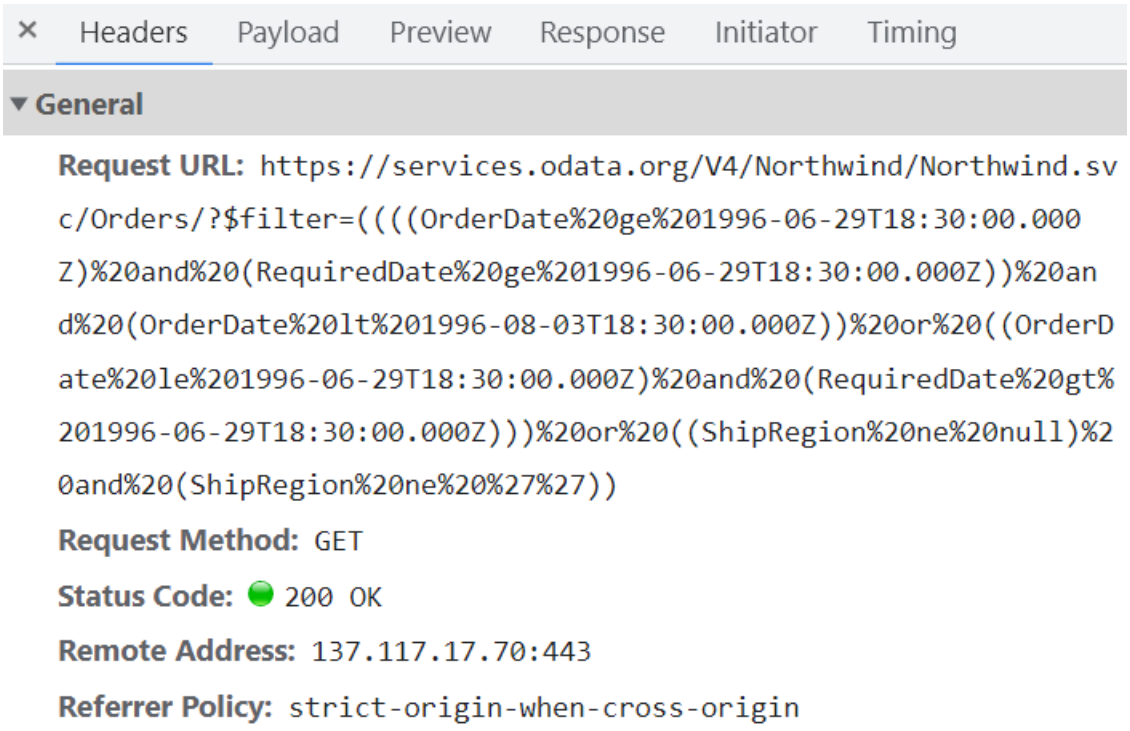
```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule height='550px' :selectedDate='selectedDate'
:eventSettings='eventSettings'
                :readOnly='readOnly'></ejs-schedule>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, View } from
'@syncfusion/ej2-vue-schedule';
Vue.use(SchedulePlugin);
let dataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/',
    adaptor: new ODataV4Adaptor(),
    crossDomain: true
});
export default {
    data() {
        return {
            selectedDate: new Date(1996, 6, 9),
            readOnly: true,
            eventSettings: {
                query: new Query(),
                includeFiltersInQuery: true,
                dataSource: dataManager, fields: {
                    id: 'Id',
                    subject: { name: 'ShipName' },
                    location: { name: 'ShipCountry' },
                    description: { name: 'ShipAddress' },
                    startTime: { name: 'OrderDate' },
                    endTime: { name: 'RequiredDate' },
                    recurrenceRule: { name: 'ShipRegion' }
                }
            }
        };
    },
    provide: {
        schedule: [Day, Week, WorkWeek, Month, Agenda]
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/data-bind-cs7" %}

The following image represents how the parameters are passed using ODataV4 filter.



Using custom adaptor

It is possible to create your own custom adaptor by extending the built-in available adaptors. The following example demonstrates the custom adaptor usage and how to add a custom field EventID for the appointments by overriding the built-in response processing using the processResponse method of the ODataV4Adaptor.

APP.VUE

```
<template>
```

```
    <div id='app'>
      <div id='container'>
        <ejs-schedule height='550px' :selectedDate='selectedDate'
:eventSettings='eventSettings' :readOnly='readOnly'></ejs-schedule>
      </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-
data';
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, View } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    class CustomAdaptor extends ODataV4Adaptor {
        processResponse() {
            let i = 0;
            // calling base class processResponse function
            let original = super.processResponse.apply(this, arguments);
            // adding employee id
            original.forEach((item) => item['EventID'] = ++i);
            return  original;
        }
    }
    let dataManager = new DataManager({
       url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/',
       adaptor: new CustomAdaptor(),
       crossDomain: true
    });
    export default {
        data() {
            return {
                selectedDate: new Date(1996, 6, 9),
                readOnly: true,
                eventSettings: { dataSource: dataManager,
                 fields: {
                        id: 'Id',
                        subject: { name: 'ShipName' },
                        location: { name: 'ShipCountry' },
                        description: { name: 'ShipAddress' },
                        startTime: { name: 'OrderDate' },
                        endTime: { name: 'RequiredDate' },
                        recurrenceRule: { name: 'ShipRegion' }
                    }
                }
            };
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda]
        }
    }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/data-bind-cs3" %}

Loading data via AJAX post

You can bind the event data through external ajax request and assign it to the dataSourceproperty of Scheduler. In the following code example, we have retrieved the data from server with the help of ajax request and assigned the resultant data to the dataSource property of Scheduler within the onSuccess event of Ajax.

[src/app/App.vue]

`html

<template>

<ejs-schedule height='550px' :selectedDate='selectedDate'

:eventSettings='eventSettings'></ejs-schedule>

</template>

<script>

import Vue from 'vue';

import { Ajax } from '@syncfusion/ej2-base';

import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, View } from '@syncfusion/ej2-vue-schedule';

Vue.use(SchedulePlugin);

let dataManager = [];

let ajax = new Ajax('Home/GetData', 'GET', false);

ajax.onSuccess = function (value) {

dataManager = value;

};

Copyright © 2001 -2024 Syncfusion Inc.

ajax.send();

export default {

data() {

return {

selectedDate: new Date(2017, 5, 11),

eventSettings: { dataSource: dataManager }

};

},

provide: {

schedule: [Day, Week, WorkWeek, Month, Agenda]

}

}

</script>

`

Definition for the controller method GetData can be referred here.

Passing additional parameters to the server

To send an additional custom parameter to the server-side post, you need to make use of the addParams method of Query. Now, assign this Query object with additional parameters to the query property of Scheduler.

APP.VUE

```
<template>
    <div id='app'>
      <div id='container'>
        <ejs-schedule height='550px' :selectedDate='selectedDate'
:eventSettings='eventSettings'
        :readOnly='readOnly'></ejs-schedule>
      </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-
data';
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, View } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    let dataManager = new DataManager({
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/',
        adaptor: new ODataV4Adaptor(),
        crossDomain: true
    });
    let dataQuery = new Query().addParams('readOnly', 'true');
```

```
    export default {
        data() {
            return {
                selectedDate: new Date(1996, 6, 9),
                readOnly: true,
                eventSettings: { dataSource: dataManager, query: dataQuery,
                fields: {
                        id: 'Id',
                        subject: { name: 'ShipName' },
                        location: { name: 'ShipCountry' },
                        description: { name: 'ShipAddress' },
                        startTime: { name: 'OrderDate' },
                        endTime: { name: 'RequiredDate' },
                        recurrenceRule: { name: 'ShipRegion' }
                    }
                }
            };
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda]
        }
    }
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/data-bind-cs4" %}

The parameters added using the query property will be sent along with the data request sent to the server on every scheduler actions.

Handling failure actions

During the time of Scheduler interacting with server, there are chances that some server-side exceptions may occur. You can acquire those error messages or exception details in client-side using the actionFailure event of Scheduler.

The argument passed to the actionFailure event contains the error details returned from the server.

APP.VUE

```
<template>
```

```
    <div id='app'>
      <div id='container'>
        <ejs-schedule ref='scheduleObj' height='530px'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:readOnly='readOnly' :actionFailure='onActionFailure'></ejs-schedule>
      </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { DataManager } from '@syncfusion/ej2-data';
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, View } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    let dataManager = new DataManager({
        url: 'http://some.com/invalidUrl'
    });
    export default {
        data() {
            return {
                selectedDate: new Date(2017, 5, 11),
                readOnly: true,
                eventSettings: { dataSource: dataManager }
            };
        },
        methods: {
            onActionFailure: function () {
                let span = document.createElement('span');
                let scheduleObj = this.$refs.scheduleObj.ej2Instances;
                scheduleObj.element.parentNode.insertBefore(span,
scheduleObj.element);
                span.style.color = '#FF0000'
                span.innerHTML = 'Server exception: 404 Not found';
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda]
        }
    }
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/data-bind-cs5" %}

The actionFailure event will be triggered not only on server returning errors, but also when there is an exception while processing any of the Scheduler CRUD actions.

Scheduler CRUD actions

The CRUD (Create, Read, Update and Delete) actions can be performed easily on Scheduler appointments using the various adaptors available within the DataManager. Most preferably, we will be using UrlAdaptor for performing CRUD actions on scheduler appointments.

[src/app/App.vue]

```
`html
<template>
<div id='app'>
<div id='container'>
<ejs-schedule height='550px' :selectedDate='selectedDate'
:eventSettings='eventSettings'></ejs-schedule>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, View } from '@syncfusion/ej2-vue-schedule';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Vue.use(SchedulePlugin);
let dataManager = new DataManager({
url: 'Home/GetData', // 'controller/actions'
crudUrl: 'Home/UpdateData',
adaptor: new UrlAdaptor
});
export default {
data() {
return {
selectedDate: new Date(2017, 5, 11),
eventSettings: { dataSource: dataManager }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
},
provide: {
schedule: [Day, Week, WorkWeek, Month, Agenda]
}
}
</script>
`
```

The server-side controller code to handle the CRUD operations are as follows.

```
`c#
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using ScheduleSample.Models;
namespace ScheduleSample.Controllers
{
public class HomeController : Controller
{
ScheduleDataDataContext db = new ScheduleDataDataContext();
public ActionResult Index()
{
return View();
}
public JsonResult LoadData()  // Here we get the Start and End Date and based on that can filter the data and return to Scheduler
{
var data = db.ScheduleEventDatas.ToList();
return Json(data, JsonRequestBehavior.AllowGet);
}
[HttpPost]
public JsonResult UpdateData(EditParams param)
```

```
{
if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code will execute while inserting the appointments
{
var value = (param.action == "insert") ? param.value : param.added[0];
int intMax = db.ScheduleEventDatas.ToList().Count > 0 ? db.ScheduleEventDatas.ToList().Max(p => p.Id) : 1;
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = new ScheduleEventData()
{
Id = intMax + 1,
StartTime = startTime.ToLocalTime(),
EndTime = endTime.ToLocalTime(),
Subject = value.Subject,
IsAllDay = value.IsAllDay,
StartTimezone = value.StartTimezone,
EndTimezone = value.EndTimezone,
RecurrenceRule = value.RecurrenceRule,
RecurrenceID = value.RecurrenceID,
RecurrenceException = value.RecurrenceException
};
db.ScheduleEventDatas.InsertOnSubmit(appointment);
db.SubmitChanges();
}
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
```

```
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appointment.StartTime = startTime.ToLocalTime();
appointment.EndTime = endTime.ToLocalTime();
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of
code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
var data = db.ScheduleEventDatas.ToList();
```

Copyright © 2001 -2024 Syncfusion Inc.

```
return Json(data, JsonRequestBehavior.AllowGet);
}
public class EditParams
{
public string key { get; set; }
public string action { get; set; }
public List<ScheduleEventData> added { get; set; }
public List<ScheduleEventData> changed { get; set; }
public List<ScheduleEventData> deleted { get; set; }
public ScheduleEventData value { get; set; }
}
}
}
`
```

Configuring Scheduler with Google API service

We have assigned our custom created Google Calendar url to the DataManager and assigned the same to the Scheduler dataSource. Since the events data retrieved from the Google Calendar will be in its own object format, therefore it needs to be resolved manually within the Scheduler's dataBinding event. Within this event, the event fields needs to be mapped properly and then assigned to the result.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :currentView='currentView'
:readonly='readonly' :eventSettings='eventSettings'
:dataBinding='onDataBinding'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
Vue.use(SchedulePlugin);
var calendarId = 'en.usa%23holiday@group.v.calendar.google.com';
var publicKey = 'AIzaSyBgbX_tgmVanBP4yafDPPXxWr70sjbKAXM';
var dataManager = new DataManager({
    url: 'https://www.googleapis.com/calendar/v3/calendars/' + calendarId +
'/events?key=' + publicKey,
    adaptor: new WebApiAdaptor,
    crossDomain: true
});
export default {
```

```
  data (){
    return {
      height: '550px',
      eventSettings: {
          dataSource: dataManager
      },
      readonly: true,
      currentView: 'Month'
    }
  },
  methods: {
      onDataBinding: function(e){
        var items = e.result.items;
        var scheduleData = [];
        if (items.length > 0) {
            for (var i = 0; i < items.length; i++) {
                var event = items[i];
                var when = event.start.dateTime;
                var start = event.start.dateTime;
                var end = event.end.dateTime;
                if (!when) {
                    when = event.start.date;
                    start = event.start.date;
                    end = event.end.date;
                }
                scheduleData.push({
                    Id: event.id,
                    Subject: event.summary,
                    StartTime: new Date(start),
                    EndTime: new Date(end),
                    IsAllDay: !event.start.dateTime
                });
            }
        }
        e.result = scheduleData;
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/data-bind-cs6" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Crud actions in Vue Schedule component

Events, a.k.a. Appointments, play an important role in Scheduler with which the users mostly interact. You can easily manipulate (add/edit/delete) the desired appointments as and when required either using the editor window or through the drag and resize action.

Add

Any kind of appointments such as normal, all-day, spanned or recurring events can be easily added on Scheduler using any one of the following ways.

- Creation using editor window
- Creation Using addEvent method

Creation using editor window

The default editor window opens when you double click on the Scheduler cells. It provides you with event related options such as Subject, Location, Start and End time, All-day, Timezone, Description and other recurrence options. With these available fields, you can choose to provide detailed information to the events. Once the fields are filled with proper values, enter the Save button to add an event.

In case, if you want to simply provide the Subject alone for appointments, just single click on the required cells which will open the quick popup expecting you to enter subject alone and save it. You can also select multiple cells and press Enter key to open the quick popup for selected time range and save the appointment for that time range.

In case, if you need to add some other additional fields to the editor window, then you can opt for custom editor window which allows you to include fields as per your application needs. If you need to add just one or two additional fields to the existing default editor window, you can do so by defining it manually and then appending it to the editor window.

Creation using addEvent method

The appointments can be created dynamically by using addEvent method. Either you can add a single or a collection of appointment objects using addEvent method. The following code example let you know how to use the addEvent method to create multiple appointments simultaneously.

APP.VUE

```
<template>
    <div id="app">
        <div id="container">
            <tr>
                <td>
                    <div>
                        <ejs-button id="btn1" v-
on:click.native="onAddClick">Add</ejs-button>
                    </div>
                </td>
            </tr>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <ejs-schedule ref='scheduleObj' id='Schedule' width="100%"
height="520px" :eventSettings='eventSettings' :selectedDate='selectedDate'>
                <e-views>
                    <e-view option="Day"></e-view>
                    <e-view option="Week"></e-view>
                    <e-view option="WorkWeek"></e-view>
                    <e-view option="Month"></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
    Vue.use(ButtonPlugin);
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
"@syncfusion/ej2-vue-schedule";
    Vue.use(SchedulePlugin);
    let scheduleData = [{
        Id: 1,
        Subject: 'Testing',
        StartTime: new Date(2018, 1, 11, 9, 0),
        EndTime: new Date(2018, 1, 11, 10, 0),
        IsAllDay: false
    },{
        Id: 2,
        Subject: 'Vacation',
        StartTime: new Date(2018, 1, 13, 9, 0),
        EndTime: new Date(2018, 1, 13, 10, 0),
        IsAllDay: false
        }];
    export default {
        data () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15),
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        },
        methods: {
            onAddClick: function () {
                let Data = [{
                        Id: 3,
                        Subject: 'Conference',
                        StartTime: new Date(2018, 1, 12, 9, 0),
                        EndTime: new Date(2018, 1, 12, 10, 0),
                        IsAllDay: true
                    },{
                        Id: 4,
                        Subject: 'Meeting',
                        StartTime: new Date(2018, 1, 15, 10, 0),
                        EndTime: new Date(2018, 1, 15, 11, 30),
                        IsAllDay: false
                        }];
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                this.$refs.scheduleObj.addEvent(Data);
            },
        }
    }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/add-event-cs1" %}

Inserting events into database at server-side

While adding the normal or recurring events to the Scheduler, insert action takes place and the following code example describes how to add a new event into database at server side.

`ts

if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code will execute while inserting the appointments

{

var value = (param.action == "insert") ? param.value : param.added[0];

int intMax = db.ScheduleEventDatas.ToList().Count > 0 ? db.ScheduleEventDatas.ToList().Max(p => p.Id) : 1;

DateTime startTime = Convert.ToDateTime(value.StartTime);

DateTime endTime = Convert.ToDateTime(value.EndTime);

ScheduleEventData appointment = new ScheduleEventData()

{

Id = intMax + 1,

StartTime = startTime.ToLocalTime(),

EndTime = endTime.ToLocalTime(),

Subject = value.Subject,

IsAllDay = value.IsAllDay,

StartTimezone = value.StartTimezone,

EndTimezone = value.EndTimezone,

RecurrenceRule = value.RecurrenceRule,

Copyright © 2001 -2024 Syncfusion Inc.

RecurrenceID = value.RecurrenceID,

RecurrenceException = value.RecurrenceException

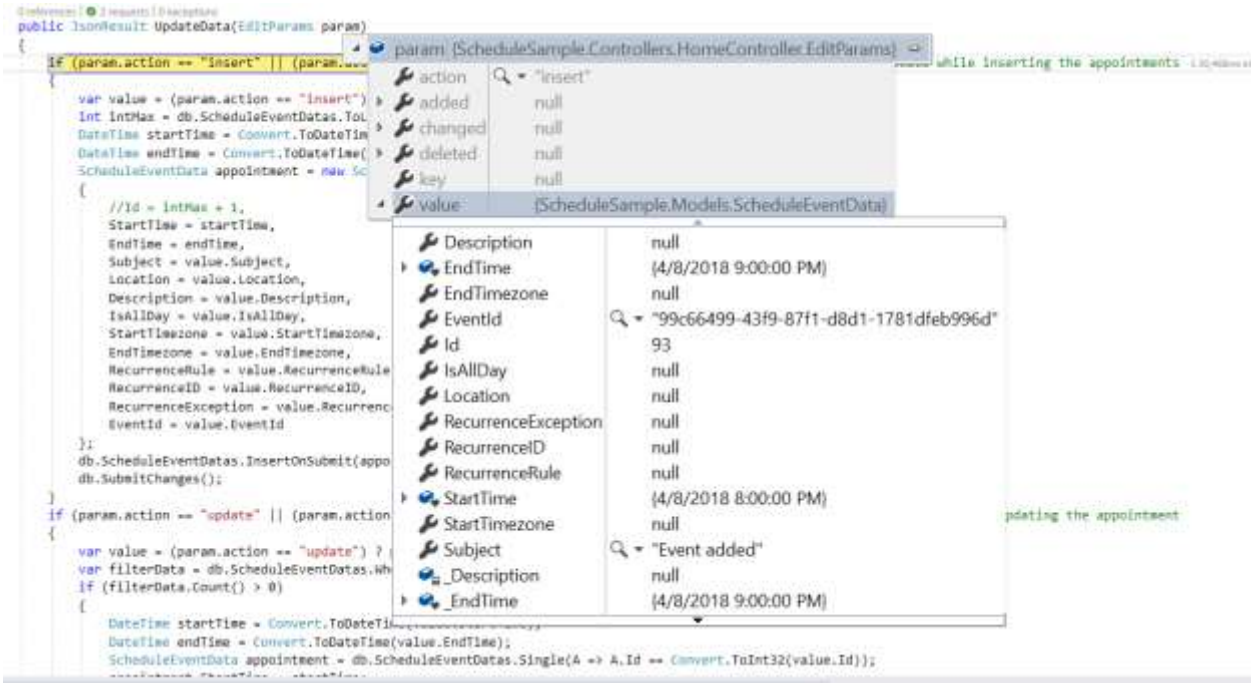
};

db.ScheduleEventDatas.InsertOnSubmit(appointment);

db.SubmitChanges();

}

`



Restricting add action based on specific criteria

In the following example, the specific fields of Scheduler editor window such as Subject and Location are made to undergo validation such that if it is left as blank, then the default Required validation message will be displayed, while clicking on a save button.

Additionally, the regex condition has been added to the Location field, so that if any special characters are typed into it, then the custom validation message will be displayed.

APP.VUE

```
<template>
    <div id="app">
        <div id="container">
            <ejs-schedule width="100%" height="550px"
:eventSettings='eventSettings' :selectedDate='selectedDate'>
                <e-views>
                    <e-view option="Day"></e-view>
                    <e-view option="Week"></e-view>
                    <e-view option="WorkWeek"></e-view>
                    <e-view option="Month"></e-view>
                    <e-view option="Agenda"></e-view>
                </e-views>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
"@syncfusion/ej2-vue-schedule";
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: {
                    dataSource: scheduleData,
                    fields: {
                        subject: { name: 'Subject', validation: { required:
true } },
                        location: {
                            name: 'Location', validation: {
                                required: true,
                                regex: ["^[a-zA-Z0-9- ]*$", 'Special
character(s) not allowed in this field']
                            }
                        }
                    },
                    selectedDate: new Date(2018, 1, 15),
                }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda]
        }
    }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/default-cs1" %}

You can also dynamically prevent the creation of appointments on Scheduler. For example, say if you want to decline the creation of appointments on weekend days, you can check for its appropriate condition within the actionBegin event.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div id="app">
        <div id="container">
            <ejs-schedule width="100%" height="550px"
:eventSettings='eventSettings' :selectedDate='selectedDate'
            :actionBegin='onActionBegin'>
                <e-views>
                    <e-view option="Day"></e-view>
                    <e-view option="Week"></e-view>
                    <e-view option="WorkWeek"></e-view>
                    <e-view option="Month"></e-view>
                    <e-view option="Agenda"></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
"@syncfusion/ej2-vue-schedule";
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: {
                    dataSource: scheduleData
                },
                selectedDate: new Date(2018, 1, 15),
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda]
        },
        methods: {
            onActionBegin: function(args) {
                let weekEnds = [0, 6];
                if(args.requestType == 'eventCreate' &&
weekEnds.indexOf((args.data[0].StartTime).getDay()) >= 0) {
                    args.cancel = true;
                }
            }
        }
    }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/default-cs2" %}

Edit

The same way the appointments such as normal, all-day, spanned or recurring events are created, it can be easily edited using any of the following ways.

- Update using editor window
- Update using saveEvent method

Update using editor window

You can open the default editor window filled with appointment details by double clicking on the required events. It gets pre-filled with event options such as Subject, Location, Start and End time, All-day, timezone, description and other recurrence options, from which you can edit the desired field values and, then enter the Save button to update it.

You can also single click on appointments, which opens the quick info popup with edit and delete options. Clicking on the edit option will open the default editor filled with event details and delete option will prompt for delete confirmation.

Update using saveEvent method

The appointments can be edited and updated manually using the saveEvent method. The following code examples shows how to edit the normal and recurring events.

Normal event - Here, an event with ID 3 is edited and its subject is changed with a new text. When the modified data object is passed onto the saveEvent method, the changes gets reflected onto the original event. The Id field is mandatory in this edit process, where the modified event object should hold the valid Id value that exists in the Scheduler data source.

APP.VUE

```
<template>
    <div id="app">
        <div id="container">
            <tr>
                <td>
                    <div>
                        <ejs-button id="btn1" v-
on:click.native="onEditClick">Edit</ejs-button>
                    </div>
                </td>
            </tr>
            <ejs-schedule ref='scheduleObj' id='Schedule' width="100%"
height="520px" :eventSettings='eventSettings' :selectedDate='selectedDate'>
                <e-views>
                    <e-view option="Day"></e-view>
                    <e-view option="Week"></e-view>
                    <e-view option="WorkWeek"></e-view>
                    <e-view option="Month"></e-view>
                </e-views>
            </ejs-schedule>
        </div>
```

```
    </div>
</template>
<script>
    import Vue from "vue";
    import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
    Vue.use(ButtonPlugin);
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
"@syncfusion/ej2-vue-schedule";
    Vue.use(SchedulePlugin);
    let scheduleData =[{
        Id: 3,
        Subject: 'Testing',
        StartTime: new Date(2018, 1, 11, 9, 0),
        EndTime: new Date(2018, 1, 11, 10, 0),
        IsAllDay: false
    },{
        Id: 4,
        Subject: 'Vacation',
        StartTime: new Date(2018, 1, 13, 9, 0),
        EndTime: new Date(2018, 1, 13, 10, 0),
        IsAllDay: false
    }];
    export default {
        data () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15),
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        },
        methods: {
            onEditClick: function () {
                let Data = {
                    Id: 3,
                    Subject: 'Testing-edited',
                    StartTime: new Date(2018, 1, 11, 10, 0),
                    EndTime: new Date(2018, 1, 11, 11, 0),
                    IsAllDay: false
                };
                this.$refs.scheduleObj.saveEvent(Data);
            }
        }
    }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/edit-event-cs1" %}

Recurring event - The following code example shows how to edit a single occurrence of a recurring event. In this case, the modified data should hold an additional field namely RecurrenceID mapping to its parent recurring event's Id value. Also, this modified occurrence will be considered as a new event in the Scheduler dataSource, where it is linked with its parent event through the RecurrenceID field value. The saveEvent method takes 2 arguments, first one accepting the modified event data object and second argument accepting either of the 2 text values - EditOccurrence or EditSeries.

When the second argument is passed as EditOccurrence, which means that the passed event data is a single modified occurrence - whereas if the second argument is passed as EditSeries, it means that the modified data needs to be edited as a whole series and therefore no new event object will be maintained in the Scheduler dataSource.

In case of modifying the single occurrence, it is also necessary to update the RecurrenceException field of parent event altogether with the occurrence editing. To know more about how to set RecurrenceException values, refer the recurring events topic.

APP.VUE

```
<template>
    <div id="app">
        <div id="container">
            <tr>
                <td>
                    <div>
                        <ejs-button id="btn1" v-
on:click.native="onEditClick">Edit</ejs-button>
                    </div>
                </td>
            </tr>
            <ejs-schedule ref='scheduleObj' width="100%" height="520px"
:eventSettings='eventSettings' :selectedDate='selectedDate'>
                <e-views>
                    <e-view option="Day"></e-view>
                    <e-view option="Week"></e-view>
                    <e-view option="WorkWeek"></e-view>
                    <e-view option="Month"></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { DataManager, Query, Predicate } from '@syncfusion/ej2-data';
    import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
    Vue.use(ButtonPlugin);
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
"@syncfusion/ej2-vue-schedule";
    Vue.use(SchedulePlugin);
    let scheduleData =[{
        Id: 3,
```

```
        Subject: 'Testing',
        StartTime: new Date(2018, 1, 11, 9, 0),
        EndTime: new Date(2018, 1, 11, 10, 0),
        IsAllDay: false,
        RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=3'
    },{
        Id: 4,
        Subject: 'Vacation',
        StartTime: new Date(2018, 1, 12, 11, 0),
        EndTime: new Date(2018, 1, 12, 12, 0),
        IsAllDay: false,
        RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2'
    }];
    export default {
        data () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15),
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        },
        methods: {
            onEditClick: function () {
                let data = new
DataManager(this.$refs.scheduleObj.getCurrentViewEvents()).executeLocal(new
Query().where('RecurrenceID', 'equal', 3));
                data[0].Subject = 'Edited';
                this.$refs.scheduleObj.saveEvent(data[0], 'EditOccurrence');
            }
        }
    }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/edit-event-cs2" %}

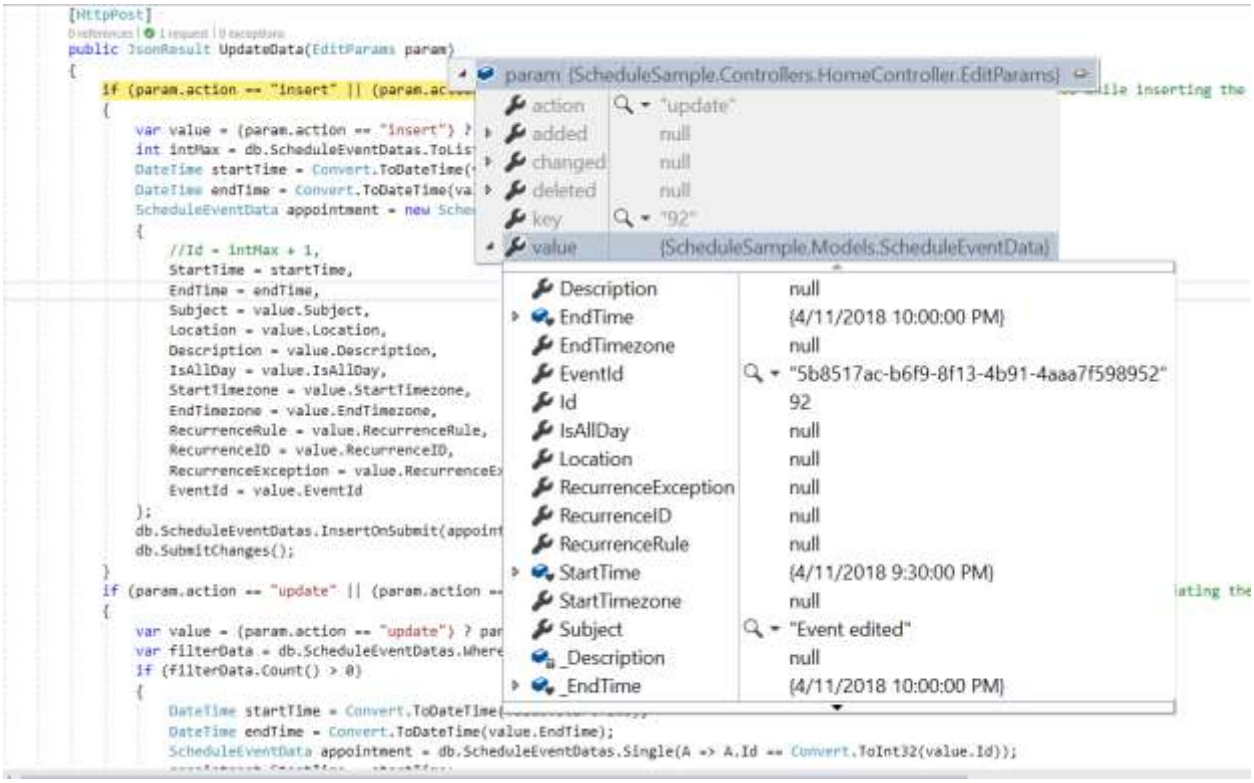
Updating events in database at server-side

While editing the normal events in the Scheduler, update action takes place and the following code example describes how to update event into database at server side.

`ts

```
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id == Convert.ToInt32(value.Id));
appointment.StartTime = startTime.ToLocalTime();
appointment.EndTime = endTime.ToLocalTime();
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
`
```

Copyright © 2001 -2024 Syncfusion Inc.



How to edit a single occurrence or entire series and update it in database at server-side

The recurring appointments can be edited in either of the following two ways.

- Single occurrence
- Entire series

Editing single occurrence - When you double click on a recurring event, a popup prompts you to choose either to edit the single event or entire series. From this, if you choose to select **EDIT EVENT** option, a single occurrence of the recurring appointment alone will be edited. The following process takes place while editing a single occurrence,

- A new event will be created from the parent event data and added to the Scheduler dataSource, with all its default field values overwritten with the newly modified data and additionally, the `recurrenceID` field will be added to it, that holds the `id` value of the parent recurring event. Also, a new `Id` will be generated for this event in the dataSource.
- The parent recurring event needs to be updated with appropriate `recurrenceException` field to hold the edited occurrence appointment's date collection.

Therefore, when a single occurrence is edited from a recurring event, the batch action takes place by allowing both the `Add` and `Edit` action requests to take place together.

In case, if you edit an existing edited occurrence of a recurring event, only those edited occurrence which present in the database as an individual event object will get updated. In this case, `update` action alone takes place on the edited occurrence object on the database.

`ts

```
if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code will execute while inserting the appointments
{
var value = (param.action == "insert") ? param.value : param.added[0];
int intMax = db.ScheduleEventDatas.ToList().Count > 0 ? db.ScheduleEventDatas.ToList().Max(p => p.Id) : 1;
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = new ScheduleEventData()
{
Id = intMax + 1,
StartTime = startTime.ToLocalTime(),
EndTime = endTime.ToLocalTime(),
Subject = value.Subject,
IsAllDay = value.IsAllDay,
StartTimezone = value.StartTimezone,
EndTimezone = value.EndTimezone,
RecurrenceRule = value.RecurrenceRule,
RecurrenceID = value.RecurrenceID,
RecurrenceException = value.RecurrenceException
};
db.ScheduleEventDatas.InsertOnSubmit(appointment);
db.SubmitChanges();
}
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
```

```
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));

appointment.StartTime = startTime.ToLocalTime();

appointment.EndTime = endTime.ToLocalTime();

appointment.StartTimezone = value.StartTimezone;

appointment.EndTimezone = value.EndTimezone;

appointment.Subject = value.Subject;

appointment.IsAllDay = value.IsAllDay;

appointment.RecurrenceRule = value.RecurrenceRule;

appointment.RecurrenceID = value.RecurrenceID;

appointment.RecurrenceException = value.RecurrenceException;

}

db.SubmitChanges();

}
`
```

Editing entire series - When you select an option **EDIT SERIES** from the popup that opens on double clicking the recurring event, the whole recurring series will be updated with the newly provided value. When this option is chosen explicitly, if a parent event holds any edited occurrences - then all its child occurrences will be removed from the dataSource and simply the single parent data will be updated.

This action of editing entire series also leads to the batch process, as both the `Delete` and `Edit` action takes place together.

```
`ts

if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of
code will execute while updating the appointment

{

var value = (param.action == "update") ? param.value : param.changed[0];

var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));

if (filterData.Count() > 0)

{

DateTime startTime = Convert.ToDateTime(value.StartTime);

DateTime endTime = Convert.ToDateTime(value.EndTime);

ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));

appointment.StartTime = startTime.ToLocalTime();

appointment.EndTime = endTime.ToLocalTime();
```

Copyright © 2001 -2024 Syncfusion Inc.

```
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
`
```

To know more about handling recurrence exceptions, refer the Adding exceptions topic.

How to edit from the current and following events of a series

The recurring appointments can be edited from current and following events when enable the property `editFollowingEvents`.

Editing Following Events - When you double click on a recurring event, a popup prompts you to choose either to edit the single event or Edit Following Events or entire series. From this, if you choose to select **EDIT FOLLOWING EVENTS** option, a current and following events of the recurring appointment will be edited. The following process takes place while editing a following events,

- A new event will be created from the parent event data and added to the Scheduler dataSource, with all its default field values overwritten with the newly modified data and additionally, the followingID field will be added to it, that holds the id value of the immediate parent recurring event. Also, a new Id will be generated for this event in the dataSource.
- The parent recurring event needs to be updated with appropriate recurrenceRule field to hold the modified occurrence appointment's end date.

Therefore, when a following events are edited from a recurring event, the batch action takes place by allowing the Add, Edit and Delete action requests to take place together.

`ts

```
if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code will execute while inserting the appointments
{
var value = (param.action == "insert") ? param.value : param.added[0];
int intMax = db.ScheduleEventDatas.ToList().Count > 0 ? db.ScheduleEventDatas.ToList().Max(p => p.Id) : 1;
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = new ScheduleEventData()
{
Id = intMax + 1,
StartTime = startTime.ToLocalTime(),
EndTime = endTime.ToLocalTime(),
Subject = value.Subject,
IsAllDay = value.IsAllDay,
StartTimezone = value.StartTimezone,
EndTimezone = value.EndTimezone,
RecurrenceRule = value.RecurrenceRule,
RecurrenceID = value.RecurrenceID,
FollowingID = value.FollowingID,
RecurrenceException = value.RecurrenceException
};
```

Copyright © 2001 -2024 Syncfusion Inc.

```
db.ScheduleEventDatas.InsertOnSubmit(appointment);
db.SubmitChanges();
}
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id == Convert.ToInt32(value.Id));
appointment.StartTime = startTime.ToLocalTime();
appointment.EndTime = endTime.ToLocalTime();
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.FollowingID = value.FollowingID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
```

Copyright © 2001 -2024 Syncfusion Inc.

```
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
`
```

Restricting edit action based on specific criteria

You can also dynamically prevent the editing of appointments on Scheduler. For example, say if you want to decline the updating of appointments on non-working hours, you can check for its appropriate condition within the actionBegin event.

APP.VUE

```
<template>
    <div id="app">
        <div id="container">
            <ejs-schedule ref='scheduleObj' width="100%" height="550px"
:eventSettings='eventSettings' :selectedDate='selectedDate'
            :workHours='workHours' :actionBegin='onActionBegin'>
                <e-views>
                    <e-view option="Day"></e-view>
                    <e-view option="Week"></e-view>
                    <e-view option="WorkWeek"></e-view>
                    <e-view option="Month"></e-view>
                    <e-view option="Agenda"></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { isNullOrUndefined } from '@syncfusion/ej2-base';
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
"@syncfusion/ej2-vue-schedule";
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
```

```
                eventSettings: {
                    dataSource: scheduleData
                },
                selectedDate: new Date(2018, 1, 15),
                workHours: {
                    highlight: true,
                    start: '11:00',
                    end: '20:00'
                }
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda]
        },
        methods: {
            onActionBegin: function(args) {
                if(args.requestType =='eventChange') {
                    let weekEnds = [0, 6];
                    let weekDay =
weekEnds.indexOf((args.data.StartTime).getDay()) >= 0;
                    let workHours =  (args.data.StartTime.getHours <
parseInt(this.$refs.scheduleObj.workHours.start)) ||
(args.data.StartTime.getHours >
parseInt(this.$refs.scheduleObj.workHours.end));
                    if(weekDay || workHours){
                        args.cancel = true;
                    }
                }
            }
        }
    }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/default-cs3" %}

Delete

The appointments can be deleted in either of the following ways,

- Selecting an appointment and clicking the delete icon from the quick popup that opens.
- Selecting an appointment and pressing Delete key.
- Selecting multiple appointments by tap holding an event and then continuously single clicking on other consecutive events and then clicking the Delete key.

Copyright © 2001 -2024 Syncfusion Inc.

- Double clicking on an event which opens the default event editor pre-filled with event details, and then choosing Delete button in it.

While performing all these above mentioned actions, a pop-up with a delete confirmation message will be displayed prompting either to proceed with deleting an appointment.

Deletion using editor window

When you double click an event, the default editor window will be opened which includes a Delete button at the bottom left position to allow you to delete that particular appointment. When deleting an appointment through this editor window, the delete alert confirmation will not be asked and the event will be deleted immediately.

Deletion using deleteEvent method

The appointments can be removed manually using the deleteEvent method. The following code examples shows how to edit the normal and recurring events.

Normal event - You can delete the normal appointments of Scheduler by simply passing its Id value or the entire event object collection to the deleteEvent method.

APP.VUE

```
<template>
    <div id="app">
        <div id="container">
            <tr>
                <td>
                    <div>
                        <ejs-button id="btn1" v-
on:click.native="onDeleteClick">Delete</ejs-button>
                    </div>
                </td>
            </tr>
            <ejs-schedule ref='scheduleObj' id='Schedule' width="100%"
height="520px" :eventSettings='eventSettings' :selectedDate='selectedDate'>
                <e-views>
                    <e-view option="Day"></e-view>
                    <e-view option="Week"></e-view>
                    <e-view option="WorkWeek"></e-view>
                    <e-view option="Month"></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
    Vue.use(ButtonPlugin);
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
"@syncfusion/ej2-vue-schedule";
    Vue.use(SchedulePlugin);
    let scheduleData = [{
        Id: 3,
        Subject: 'Testing',
        StartTime: new Date(2018, 1, 11, 9, 0),
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        EndTime: new Date(2018, 1, 11, 10, 0),
        IsAllDay: false
    },{
        Id: 4,
        Subject: 'Vacation',
        StartTime: new Date(2018, 1, 13, 9, 0),
        EndTime: new Date(2018, 1, 13, 10, 0),
        IsAllDay: false
        }];
    export default {
        data () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15),
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        },
        methods: {
            onDeleteClick: function () {
                this.$refs.scheduleObj.deleteEvent(4);
            }
        }
    }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/delete-event-cs1" %}

Recurring Event - The recurring events can be removed as an entire series or simply removing single occurrence by using the deleteEvent method which takes in either the DeleteSeries or DeleteOccurrence parameters. The following code example shows how to delete entire series.

APP.VUE

```
<template>
    <div id="app">
        <div id="container">
            <tr>
                <td>
                    <div>
                        <ejs-button id="btn1" v-
on:click.native="onDeleteClick">Delete</ejs-button>
                    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                </td>
            </tr>
            <ejs-schedule ref='scheduleObj' id='Schedule' width="100%"
height="520px" :eventSettings='eventSettings' :selectedDate='selectedDate'>
                <e-views>
                    <e-view option="Day"></e-view>
                    <e-view option="Week"></e-view>
                    <e-view option="WorkWeek"></e-view>
                    <e-view option="Month"></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
    Vue.use(ButtonPlugin);
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
"@syncfusion/ej2-vue-schedule";
    Vue.use(SchedulePlugin);
    let scheduleData = [{
        Id: 3,
        Subject: 'Testing',
        StartTime: new Date(2018, 1, 11, 9, 0),
        EndTime: new Date(2018, 1, 11, 10, 0),
        IsAllDay: false,
        RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=3'
    },{
        Id: 4,
        Subject: 'Vacation',
        StartTime: new Date(2018, 1, 12, 11, 0),
        EndTime: new Date(2018, 1, 12, 12, 0),
        IsAllDay: false,
        RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2'
        }];
    export default {
        data () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15),
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        },
        methods: {
            onDeleteClick: function () {
                let data = [{
                    Id: 4,
                    Subject: 'Vacation',
                    RecurrenceID: 4,
                    StartTime: new Date(2018, 1, 12, 11, 0),
                    EndTime: new Date(2018, 1, 12, 12, 0),
                    IsAllDay: false,
                    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2'
                }];
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                this.$refs.scheduleObj.deleteEvent(data,'DeleteSeries');
            },
        }
    }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/delete-event-cs2" %}

Removing events from database at server-side

While deleting the event from the Scheduler, remove action takes place and the following code example describes how to delete event from database at server side.

`ts

if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of code will execute while removing the appointment

{

if (param.action == "remove")

{

int key = Convert.ToInt32(param.key);

ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();

if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);

}

else

{

foreach (var apps in param.deleted)

{

ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();

if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);

}

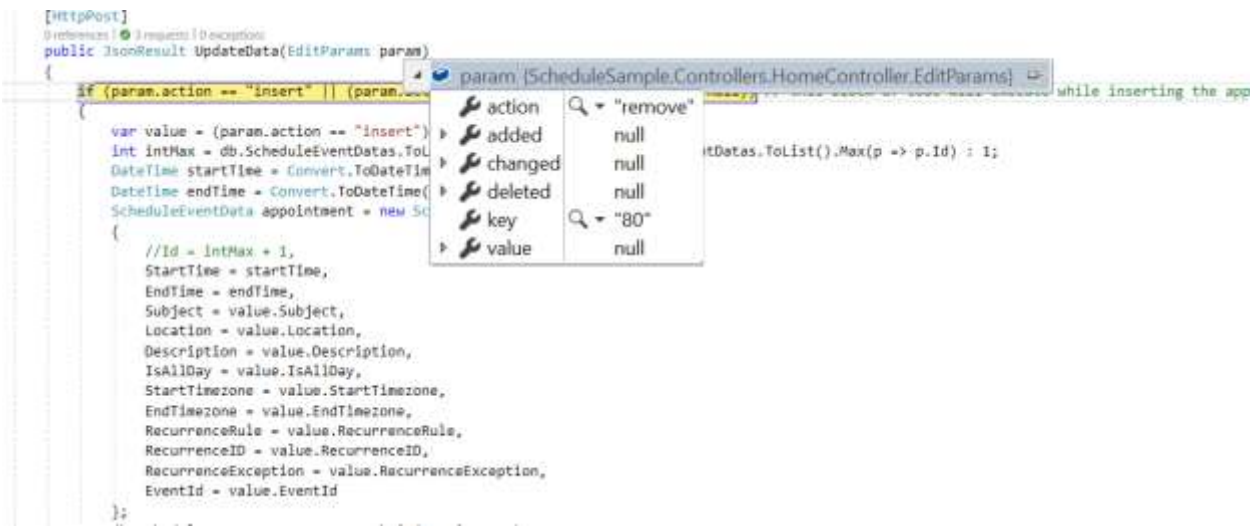
}

db.SubmitChanges();

Copyright © 2001 -2024 Syncfusion Inc.

}

`



How to delete a single occurrence or entire series from Scheduler and update it in database at server-side

The recurring events can be deleted in either of the following two ways.

- Single occurrence
- Entire series

Single occurrence - When you attempt to delete the recurring events, a popup prompts you to choose either to delete the single event or entire series. From this, if you choose to select **DELETE EVENT** option, a single occurrence of the recurring appointment alone will be removed. The following process takes place while removing a single occurrence,

- The selected occurrence will be deleted from the Scheduler user interface.
- In code, the parent recurring event object will be updated with appropriate `recurrenceException` field, to hold the deleted occurrence appointment's date collection.

Therefore, when a single occurrence is deleted from a recurring event, the `update` action takes place on the parent recurring event as shown in the following code example.

In case, if you delete an existing edited occurrence of a recurring event, only those edited occurrence which present in the database as an individual event object will get removed. In this case, `delete` action takes place instead of `update` action and the parent recurring event object remains same with no changes.

`ts

if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of code will execute while updating the appointment

{

var value = (param.action == "update") ? param.value : param.changed[0];

var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));

Copyright © 2001 -2024 Syncfusion Inc.

```
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appointment.StartTime = startTime.ToLocalTime();
appointment.EndTime = endTime.ToLocalTime();
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
`
```

Entire series - When you select an option **DELETE SERIES** from the popup, the whole recurring series will be deleted. When this option is chosen explicitly, if a parent event holds any edited occurrences - then all its child occurrences which are maintained as separate event objects will also be removed from the dataSource. This action of deleting entire series leads to remove action and removes one or more event objects at the same time.

```ts
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of
code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
```

```
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
`
```

How to delete only the current and following events of a series

The recurring events can be deleted from current and following events only when enable `editFollowingEvents` property.

Delete Following Events - When you attempt to delete the recurring events, a popup prompts you to choose either to delete the single event or Following Events or entire series. From this, if you choose to select **FOLLOWING EVENT** option, a current and following events of the recurring appointment alone will be removed. The following process takes place while removing a single occurrence,

- The selected occurrence and the following events in same series will be deleted from the Scheduler user interface.
- In code, the parent recurring event object will be updated with appropriate `recurrenceRule` field, to update the end date of the recurring events.

Therefore, when following events are deleted from a recurring event, the `remove` and `update` action takes place on the immediate parent recurring event as shown in the following code example.

```
`ts
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id == Convert.ToInt32(value.Id));
appointment.StartTime = startTime.ToLocalTime();
```

Copyright © 2001 -2024 Syncfusion Inc.

```
appointment.EndTime = endTime.ToLocalTime();
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.FollowingID = value.FollowingID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
`
```

Copyright © 2001 -2024 Syncfusion Inc.

Drag and drop

When you drag and drop a normal event on the Scheduler, the event editing action takes place. When a recurring event is drag and dropped on a desired time range, the batch action explained in Editing a single occurrence process will takes place - thus allowing both the Add and Link to the Video action to take place together.

By default, when you drag a recurring instance, only the occurrence of the event gets edited and not a whole series.

You can watch the following video to learn more about Vue Scheduler's advanced drag and resize options:

APP.VUE

```
<template>
    <div id="app">
        <div id="container">
            <ejs-schedule ref='scheduleObj' id='Schedule' width="100%"
height="550px" :eventSettings='eventSettings' :selectedDate='selectedDate'>
                <e-views>
                    <e-view option="Day"></e-view>
                    <e-view option="Week"></e-view>
                    <e-view option="WorkWeek"></e-view>
                    <e-view option="Month"></e-view>
                    <e-view option="Agenda"></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, DragAndDrop
} from "@syncfusion/ej2-vue-schedule";
    import { scheduleData } from './datasource.js'
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15),
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda, DragAndDrop]
        }
    }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs38" %}

Resize

When you resize a normal event on the Scheduler, the event editing action takes place. When a recurring event is resized to a new desired time, the batch action explained in Editing a single occurrence process will takes place - thus allowing both the Add and Edit action to take place together.

By default, when you resize a recurring instance, only the occurrence of the event gets edited and not a whole series.

APP.VUE

```
<template>
    <div id="app">
        <div id="container">
            <ejs-schedule ref='scheduleObj' id='Schedule' width="100%"
height="550px" :eventSettings='eventSettings' :selectedDate='selectedDate'>
                <e-views>
                    <e-view option="Day"></e-view>
                    <e-view option="Week"></e-view>
                    <e-view option="WorkWeek"></e-view>
                    <e-view option="Month"></e-view>
                    <e-view option="Agenda"></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, Resize }
from "@syncfusion/ej2-vue-schedule";
    import { scheduleData } from './datasource.js'
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15),
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda, Resize]
        }
    }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/event-cs39" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Virtual scrolling in Vue Schedule component

To achieve better performance in the Scheduler when loading a large number of resources and events, we have added virtual scrolling support load a large set of resources and events instantly as you scroll. You can dynamically load large number of resources and events in the Scheduler by setting true to the allowVirtualScrolling property within the view specific settings. The virtual loading of events is possible in Agenda view, by setting allowVirtualScrolling property to true within the agenda view specific settings.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' ref='ScheduleObj' width='100%'
height='550px' :selectedDate='selectedDate' :eventSettings='eventSettings'
:group='group'>
                <e-views>
                    <e-view option='TimelineMonth'
:eventTemplate='timelineEventTemplate'
:allowVirtualScrolling='virtualScroll'  isSelected="true"></e-view>
                     <e-view option="TimelineYear" orientation="Vertical"
:allowVirtualScrolling="virtualScroll"></e-view>
                </e-views>
                <e-resources>
                    <e-resource field='OwnerId' title='Owner' name='Owners'
:allowMultiple='allowMultiple' :dataSource='ownerData' textField='Text'
idField='Id' colorField='Color'>
                    </e-resource>
                </e-resources>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { generateResourceData, generateStaticEvents } from
'./datasource.js';
    import { SchedulePlugin, TimelineViews, TimelineMonth, TimelineYear,
Resize, DragAndDrop } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    var timelineEventTemplateVue = Vue.component('timelineTemplate', {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        template: `<div class='template-wrap' style='{background:
data.PrimaryColor}'>
            <div class="subject" style='{background:
data.SecondaryColor};'>{{data.Subject}}</div></div>`,
        data() {
            return {
                data: {}
            };
        }
    });
    export default {
        data: function () {
            return {
                selectedDate: new Date(2018, 4, 1),
                timelineEventTemplate: function (e) {
                    return {
                        template: timelineEventTemplateVue
                    };
                },
                allowMultiple: true,
                virtualScroll: true,
                group: {
                    byGroupID: false,
                    resources: ['Owners']
                },
                ownerData: generateResourceData(1, 300, 'Resource'),
                eventSettings: { dataSource: generateStaticEvents(new
Date(2018, 4, 1), 300, 12) }
            }
        },
        provide: {
            schedule: [TimelineViews, TimelineMonth, TimelineYear, Resize,
DragAndDrop]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
    .e-schedule .e-timeline-month-view .template-wrap .subject {
        padding: 10px 25px;
    }
    .e-schedule .e-timeline-year-view .template-wrap .subject {
        padding: 1px 25px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
    .e-schedule .e-more-event-popup .template-wrap .subject {
        padding: 0px 25px;
    }
    .e-schedule .template-wrap {
        width: 100%;
    }
    .e-schedule .e-timeline-month-view .e-resource-left-td {
        width: 150px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/virtual-scrolling-cs2" %}

For now, the virtual loading of resources and events is not supported in MonthAgenda, Year and TimelineYear (Horizontal Orientation) views.

Enabling lazy loading for appointments

The lazy loading feature provides a convenient way to efficiently load resource appointments into the Scheduler using an on-demand approach. With this feature, you can seamlessly load a large volume of appointment data into the Scheduler without experiencing any performance degradation.

By default, the Scheduler fetches all the relevant appointments from the server with in the current date range. However, enabling this feature will trigger query requests to the server for appointment retrieval whenever new resources are rendered due to scroll actions. These queries contain the resource IDs of currently displayed resources along with current date range, which can be passed as a comma-separated string. In the server controller, these resource IDs are parsed to filter the necessary appointments to render in the scheduler.

When you enable this feature, the Scheduler becomes capable of fetching events from remote services only for the current view port alone to optimize the data retrieval. The remaining appointment data is fetched form the server on-demand based on currently rendered view port resources as you scroll's through the scheduler content.

To enable this feature, you have to set the enableLazyLoading property to true within the view specific settings.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' ref='ScheduleObj' width='100%'
height='550px' :selectedDate='selectedDate' readonly='true'
:eventSettings='eventSettings' :group='group'>
                <e-views>
                    <e-view option='TimelineMonth'
:enableLazyLoading='enableLazyLoad' isSelected="true"></e-view>
                </e-views>
                <e-resources>
                    <e-resource field='ResourceId' title='Resource'
name='Resources' :dataSource='resourceData' textField='Text' idField='Id'
colorField='Color'>
                    </e-resource>
                </e-resources>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { generateResourceData } from './datasource.js';
    import { SchedulePlugin, TimelineMonth } from '@syncfusion/ej2-vue-
schedule';
    import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
    Vue.use(SchedulePlugin);
    var dataManager = new DataManager({
        url:
'https://services.syncfusion.com/vue/production/api/VirtualEventData',
        adaptor: new WebApiAdaptor,
        crossDomain: true
    });
    export default {
        data: function () {
            return {
                selectedDate: new Date(2023, 3, 1),
                enableLazyLoad: true,
                group: {
                    resources: ['Resources']
                },
                resourceData: generateResourceData(1, 1000, 'Resource'),
                eventSettings: { dataSource: dataManager }
            }
        },
        provide: {
            schedule: [TimelineMonth]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/lazy-loading-cs1" %}

Here's the server-side controller code that retrieves appointment data based on the resource IDs provided as query parameters:

```c#
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.OData.Query;
namespace LazyLoadingServices.Controllers
{
public class VirtualEventDataController : Controller
{
private readonly EventsContext dbContext;
[HttpGet]
[EnableQuery]
[Route("api/VirtualEventData")]
public IActionResult GetData([FromQuery] Params param)
{
IQueryable<EventData> query = dbContext.Events;
// Filter the appointment data based on the ResourceId query params.
if (!string.IsNullOrEmpty(param.ResourceId))
{
string[] resourceId = param.ResourceId.Split(',');
query = query.Where(data => resourceId.Contains(data.ResourceId.ToString()));
}
return Ok(query.ToList());
}
}
public class Params
{
public DateTime? StartDate { get; set; }
public DateTime? EndDate { get; set; }
public string ResourceId { get; set; }
```

}

}

`

Note:

- The property will be effective, when large number of resources and appointments bound to the Scheduler.
- This property is applicable only when resource grouping is enabled in Scheduler.

See Also

- Virtual scrolling in Agenda view

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Editor template in Vue Schedule component

Scheduler makes use of popups and dialog to display the required notifications, as well as includes an editor window with event fields for making the appointment creation and editing process easier. You can also easily customize the editor window and the fields present in it, and can also apply validations on those fields.

Event editor

The editor window usually opens on the Scheduler, when a cell or event is double clicked. When a cell is double clicked, the detailed editor window opens in "Add new" mode, whereas when an event is double clicked, the same is opened in an "Edit" mode.

In mobile devices, you can open the detailed editor window in edit mode by clicking the edit icon on the popup, that opens on single tapping an event. You can also open it in add mode by single tapping a cell, which will display a `+` indication, clicking on it again will open the editor window.

You can also prevent the editor window from opening, by rendering Scheduler in a `readonly` mode or by doing code customization within the popupOpen event.

How to change the editor window header title and text of footer buttons

You can change the header title and the text of buttons displayed at the footer of the editor window by changing the appropriate localized word collection used in the Scheduler.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :views='views'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { L10n } from '@syncfusion/ej2-base';
import { SchedulePlugin, Day, Week, TimelineViews, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
L10n.load({
    'en-US': {
        'schedule': {
            'saveButton': 'Add',
            'cancelButton': 'Close',
            'deleteButton': 'Remove',
            'newEvent': 'Add Event',
        },
    }
});
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['TimelineDay', 'Day', 'Week', 'Month', 'Agenda'],
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, TimelineViews, Month, Agenda]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs1" %}

How to change the label text of default editor fields

To change the default labels such as Subject, Location and other field names in the editor window, make use of the `title` property available within the field option of `eventSettings`.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :views='views'
:eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, TimelineViews, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['TimelineDay', 'Day', 'Week', 'Month', 'Agenda'],
      eventSettings: {
          dataSource: scheduleData,
          fields: {
            id: 'Id',
            subject: { name: 'Subject', title: 'Event Name' },
            location: { name: 'Location', title: 'Event Location'},
            description: { name: 'Description', title: 'Event Description'
},
            startTime: { name: 'StartTime', title: 'Start Duration' },
            endTime: { name: 'EndTime', title: 'End Duration'  }
        }
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, TimelineViews, Month, Agenda]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs2" %}

Field validation

It is possible to validate the required fields of the editor window from client-side before submitting it, by adding appropriate validation rules to each field. The appointment fields have been extended to accept both string and object type values. To perform validations, it is necessary to specify object values for the event fields.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
let minValidation: (args) =>  = (args) => {
    return args['value'].length >= 5;
};
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      eventSettings: {
          dataSource: scheduleData,
          fields: {
            id: 'Id',
            subject: { name: 'Subject', validation: { required: true } },
            location: { name: 'Location', validation: { required: true } },
            description: {
                name: 'Description', validation: {
                    required: true, minLength: [minValidation, 'Need atleast
5 letters to be entered']
                }
            },
            startTime: { name: 'StartTime', validation: { required: true }
},
            endTime: { name: 'EndTime', validation: { required: true } }
          }
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
```

```
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs3" %}

Applicable validation rules can be referred from form validation documentation.

Add additional fields to the default editor

The additional fields can be added to the default event editor by making use of the popupOpen event which gets triggered before the event editor opens on the Scheduler. The popupOpen is a client-side event that triggers before any of the generic popups opens on the Scheduler. The additional field (any of the form elements) should be added with a common class name e-field, so as to handle and process those additional data along with the default event object. In the following example, an additional field Event Type has been added to the default event editor and its value is processed accordingly.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width' :views='views'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:popupOpen='onPopupOpen'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
vue-schedule';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { createElement } from '@syncfusion/ej2-base';
import { eventData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
```

```
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
          dataSource: eventData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onPopupOpen: function(args) {
        if (args.type === 'Editor') {
            // Create required custom elements in initial time
            if (!args.element.querySelector('.custom-field-row')) {
              let row = createElement('div', { className: 'custom-field-row'
});
              let formElement = args.element.querySelector('.e-schedule-
form');
              formElement.firstChild.insertBefore(row,
args.element.querySelector('.e-title-location-row'));
              let container = createElement('div', { className: 'custom-
field-container' });
              let inputEle = createElement('input', {
                  className: 'e-field', attrs: { name: 'EventType' }
              });
              container.appendChild(inputEle);
              row.appendChild(container);
              let dropDownList = new DropDownList({
                  dataSource: [
                    { text: 'Public Event', value: 'public-event' },
                    { text: 'Maintenance', value: 'maintenance' },
                    { text: 'Commercial Event', value: 'commercial-event' },
                    { text: 'Family Event', value: 'family-event' }
                  ],
                    fields: { text: 'text', value: 'value' },
                    value: args.data.EventType,
                    floatLabelType: 'Always', placeholder: 'Event Type'
              });
              dropDownList.appendTo(inputEle);
              inputEle.setAttribute('name', 'EventType');
            }
        }
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";

  .custom-field-row {
    padding-bottom: 14px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs4" %}

Customizing the default time duration in editor window

In default event editor window, start and end time duration are processed based on the interval value set within the timeScale property. By default, interval value is set to 30, and therefore the start/end time duration within the event editor will be in a 30 minutes time difference. You can change this duration value by changing the duration option within the popupOpen event as shown in the following code example.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width' :views='views'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:popupOpen='onPopupOpen'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
          dataSource: scheduleData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onPopupOpen: function(args) {
        if (args.type === 'Editor') {
```

```
            args.duration = 60;
          }
      }
    },
    provide: {
      schedule: [Day, Week, WorkWeek, Month]
    }
}
</script>
<style>
    @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs5" %}

How to prevent the display of editor and quick popups

It is possible to prevent the display of editor and quick popup windows by passing the value `true` to `cancel` option within the popupOpen event.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width' :views='views'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:popupOpen='onPopupOpen'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      eventSettings: {
          dataSource: scheduleData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onPopupOpen: function(args) {
        args.cancel = true;
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs6" %}

In case, if you need to prevent only specific popups on Scheduler, then you can check the condition based on the popup type. The types of the popup that can be checked within the popupOpen event are as follows.

| Type | Description |

|------|-------------|

| Editor | For Detailed editor window.|

| QuickInfo | For Quick popup which opens on cell click.|

| EditEventInfo |For Quick popup which opens on event click.|

| ViewEventInfo | For Quick popup which opens on responsive mode.|

| EventContainer | For more event indicator popup.|

| RecurrenceAlert | For edit recurrence event alert popup.|

| DeleteAlert | For delete confirmation popup.|

Copyright © 2001 -2024 Syncfusion Inc.

| ValidationAlert | For validation alert popup.|

| RecurrenceValidationAlert | For recurrence validation alert popup.|

Customizing timezone collection in the editor window

By default, the timezone collections in the editor window have been loaded with built-in timezone collections. Now we can be able to customize the timezone collections using the timezoneDataSource property with the collection of TimezoneFields data.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate' :views =
'views' :eventSettings='eventSettings'
:timezoneDataSource='timezoneDataSource'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import {SchedulePlugin, Day, Week, Month, Agenda } from '@syncfusion/ej2-
vue-schedule';
let data = [{
        Id: 1,
        Subject: 'Explosion of Betelgeuse Star',
        StartTime: new Date(2020, 1, 15, 10, 0),
        EndTime: new Date(2018, 1, 15, 12, 30),
        IsAllDay: false
    },{
        Id: 2,
        Subject: 'Blue Moon Eclipse',
        StartTime: new Date(2020, 1, 16, 12, 0),
        EndTime: new Date(2018, 1, 16, 13, 0),
        IsAllDay: false
    }];
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      views: ['Day', 'Week', 'Month', 'Agenda'],
      eventSettings: {dataSource: data},
      timezoneDataSource : [
                    { Value: 'Pacific/Niue', Text: 'Niue' },
                    { Value: 'Pacific/Pago_Pago', Text: 'Pago Pago' },
                    { Value: 'Pacific/Honolulu', Text: 'Hawaii Time' },
                    { Value: 'Pacific/Rarotonga', Text: 'Rarotonga' },
                    { Value: 'Pacific/Tahiti', Text: 'Tahiti' },
                ],
      selectedDate: new Date(2020, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, Month, Agenda]
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs7" %}

Customizing event editor using template

The event editor window can be customized by making use of the editorTemplate option. Here, the custom window design is built with the required fields using the script template and its type should be of **text/x-template**.

Each field defined within template should contain the **e-field** class, so as to allow the processing of those field values internally. The ID of this customized script template section is assigned to the editorTemplate option, so that these customized fields will be replaced onto the default editor window.

Note: **e-field** class only applicable for **DropDownList**, **DateTimePicker**, **MultiSelect**, **DatePicker**, **CheckBox** and **TextBox** components. Since we have processed the field values internally for the above mentioned components.

As we are using our Syncfusion sub-components within our editor using template in the following example, the custom defined form elements needs to be configured as required Syncfusion components such as **DropDownList** and **Link to the Video** within the popupOpen event. This particular step can be skipped, if the user needs to simply use the usual form elements.

Check out this video for further information on how to customise an editor window using a template in the Vue Scheduler:

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
      <ejs-schedule :height='height' :width='width' :views='views'
:selectedDate='selectedDate'
        :eventSettings='eventSettings' :editorTemplate='editorTemplate'
:popupOpen='onPopupOpen'>
      </ejs-schedule>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
vue-schedule';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { DateTimePicker } from '@syncfusion/ej2-calendars';
import { eventData } from './datasource.js';
Vue.use(SchedulePlugin);
var editorTemplateVue = Vue.component('editorTemplate', {
  template: `<table class="custom-event-editor" width="100%"
cellpadding="5"><tbody><tr>
  <td class="e-textlabel">Summary</td><td colspan="4"><input id="Subject"
class="e-field e-input" type="text" value="" name="Subject" style="width:
100%" /></td></tr><tr><td class="e-textlabel">Status</td><td
colspan="4"><input type="text" id="EventType" name="EventType" class="e-
field" style="width: 100%" /></td></tr><tr><td class="e-
textlabel">From</td><td colspan="4"><input id="StartTime" class="e-field"
type="text" name="StartTime" /></td></tr><tr>
  <td class="e-textlabel">To</td><td colspan="4"><input id="EndTime"
class="e-field" type="text" name="EndTime" /></td></tr><tr><td class="e-
textlabel">Reason</td><td colspan="4">
  <textarea id="Description" class="e-field e-input" name="Description"
rows="3" cols="50"                 style="width: 100%; height: 60px
!important; resize: vertical"></textarea></td></tr></tbody></table>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data() {
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
        dataSource: eventData
      },
      selectedDate: new Date(2018, 1, 15),
      showQuickInfo: false,
      editorTemplate: function(e) {
        return {
          template: editorTemplateVue
        };
      }
    };
  },
  methods: {
    onPopupOpen: function(args) {
      if (args.type === 'Editor') {
        let statusElement = args.element.querySelector('#EventType');
        if (!statusElement.classList.contains('e-dropdownlist')) {
          let dropDownListObject = new DropDownList({
            placeholder: 'Choose status',
```

```
            value: statusElement.value,
            dataSource: ['New', 'Requested', 'Confirmed']
          });
          dropDownListObject.appendTo(statusElement);
          statusElement.setAttribute('name', 'EventType');
        }
        let startElement = args.element.querySelector('#StartTime');
        if (!startElement.classList.contains('e-datetimepicker')) {
          new DateTimePicker(
            { value: new Date(startElement.value) || new Date() },
            startElement
          );
        }
        let endElement = args.element.querySelector('#EndTime');
        if (!endElement.classList.contains('e-datetimepicker')) {
          new DateTimePicker(
            { value: new Date(endElement.value) || new Date() },
            endElement
          );
        }
      }
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month]
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
.custom-event-editor .e-textlabel {
  padding-right: 15px;
  text-align: right;
}
.custom-event-editor td {
  padding: 7px;
  padding-right: 16px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

How to customize header and footer using template

The editor window's header and footer can be enhanced with custom designs using the editorHeaderTemplate and editorFooterTemplate options. To achieve this, create a script template that includes the necessary fields. Ensure that the template type is set to **text/x-template**.

In this demo, we tailor the editor's header according to the appointment's subject field using the editorHeaderTemplate. Furthermore, we make use of the editorFooterTemplate to handle the functionality of validating specific fields before proceeding with the save action or canceling it if validation requirements are not met.

APP.VUE

```
<template>
  <div id="app">
    <div id="container">
      <ejs-schedule
        :height="height"
        :width="width"
        ref='schedule_instance'
        :eventSettings="eventSettings"
        :editorHeaderTemplate="editorHeaderTemplate"
        :editorFooterTemplate="editorFooterTemplate"
        :popupOpen="onPopupOpen"
      >
      </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import {
  SchedulePlugin,
  Day,
  Week,
  WorkWeek,
  Month,
} from "@syncfusion/ej2-vue-schedule";
Vue.use(SchedulePlugin);
const today = new Date();
var data = [
  {
    Id: 1,
    Subject: "Surgery - Andrew",
    StartTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 9, 0),
    EndTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 10, 0),
    IsAllDay: false
  },
  {
    Id: 2,
    Subject: "Consulting - John",
    StartTime: new Date(today.getFullYear(),
today.getMonth(),today.getDate(), 10, 0),
    EndTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 11, 30),
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    IsAllDay: false
  },
  {
    Id: 3,
    Subject: "Therapy - Robert",
    StartTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 11,30),
    EndTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 12, 30),
    IsAllDay: false
  },
  {
    Id: 4,
    Subject: "Observation - Steven",
    StartTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(),12, 30),
    EndTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 13, 30),
    IsAllDay: false
  },
];
var editorHeaderTemplateVue = Vue.component("editorHeaderTemplate", {
  template: `<div class="event-header">{{getName(data)}}</div></div>`,
  data: () => ({}),
  methods: {
    getName: function(value) {
      return value.Subject ? value.Subject : 'Create New Event';
    }
  }
});
var editorFooterTemplateVue = Vue.component("editorFooterTemplate", {
  template: `<div id="event-footer"><div id="verify"><input type="checkbox"
id="check-box" value="unchecked" />
  <label htmlFor="check-box" id="text">Verified</label></div><div id="right-
button">
  <button id="Save" className="e-control e-btn e-primary" disabled data-
ripple="true"> Save</button>
  <button id="Cancel" className="e-control e-btn e-primary" data-
ripple="true"> Cancel </button></div></div>`,
  data() {
    return {
      data: {},
    };
  },
});
export default {
  data() {
    return {
      height: "550px",
      width: "100%",
      eventSettings: {dataSource: data},
      editorHeaderTemplate: function () {
        return {
          template: editorHeaderTemplateVue,
        };
      },
      editorFooterTemplate: function () {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        return {
          template: editorFooterTemplateVue,
        };
      },
    };
  },
  methods: {
    onPopupOpen: function (args) {
      if (args.type === "Editor") {
        let scheduleObj = this.$refs.schedule_instance.ej2Instances
        const saveButton = args.element.querySelector("#Save");
        const cancelButton = args.element.querySelector("#Cancel");
        const checkBox = args.element.querySelector("#check-box");
        checkBox.onchange = () => {
          if (!checkBox.checked) {
            saveButton.setAttribute("disabled", "");
          } else {
            saveButton.removeAttribute("disabled");
          }
        };
        saveButton.onclick = () => {
          const data = {
            Id: args.data.Id,
            Subject: args.element.querySelector("#Subject").value,
            StartTime:
args.element.querySelector("#StartTime").ej2_instances[0].value,
            EndTime:
args.element.querySelector("#EndTime").ej2_instances[0].value,
            IsAllDay: args.element.querySelector("#IsAllDay").checked,
          };
          if (args.target?.classList.contains("e-appointment")) {
            scheduleObj.saveEvent(data, "Save");
          } else {
            data.Id = scheduleObj.getEventMaxID();
            scheduleObj.addEvent(data);
          }
          scheduleObj.closeEditor();
        };
        cancelButton.onclick = () => {
          scheduleObj.closeEditor();
        };
      }
    },
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month],
  },
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
#verify {
   position: fixed;
   padding: 0 20px;
}
#text {
   cursor: pointer;
   display: inline-block;
   font-family: "Roboto", -apple-system, BlinkMacSystemFont, "Segoe UI",
"Helvetica Neue", sans-serif;
   font-size: 14px;
   font-weight: normal;
   line-height: 14px;
   user-select: none;
   margin-left: 8px;
   vertical-align: middle;
   white-space: normal;
}
#right-button {
   padding: 0 10px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-header-footer-cs1" %}

How to add resource options within editor template

The resource field can be added within editor template with multiselect control for allow multiple resources.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
      <ejs-schedule :height='height' :width='width' :views='views'
:selectedDate='selectedDate'
        :eventSettings='eventSettings' :editorTemplate='editorTemplate'
:popupOpen='onPopupOpen'
        :group='group'>
        <e-resources>
          <e-resource field='OwnerId' title='Owner' name='Owners'
:dataSource='ownerDataSource' textField='text' idField='id'
colorField='color'>
          </e-resource>
        </e-resources>
      </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { SchedulePlugin, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
vue-schedule';
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
import { DateTimePicker } from '@syncfusion/ej2-calendars';
import { eventData } from './datasource.js';
Vue.use(SchedulePlugin);
var editorTemplateVue = Vue.component('editorTemplate', {
  template: `<table class="custom-event-editor" width="100%"
cellpadding="5">
        <tbody>
            <tr>
                <td class="e-textlabel">Summary</td>
                <td colspan="4">
                    <input id="Subject" class="e-field e-input" type="text"
value="" name="Subject" style="width: 100%" />
                </td>
            </tr>
            <tr>
            <td class="e-textlabel">From</td>
                <td colspan="4">
                    <input id="StartTime" class="e-field" type="text"
name="StartTime" />
                </td>
            </tr>
            <tr>
                <td class="e-textlabel">To</td>
                <td colspan="4">
                    <input id="EndTime" class="e-field" type="text"
name="EndTime" />
                </td>
            </tr>
            <tr>
                <td class="e-textlabel">Owner</td>
                <td colspan="4">
                    <input type="text" id="OwnerId" name="OwnerId" class="e-
field" style="width: 100%" />
                </td>
            </tr>
            <tr>
                <td class="e-textlabel">Reason</td>
                <td colspan="4">
                    <textarea id="Description" class="e-field e-input"
name="Description" rows="3" cols="50" style="width: 100%; height: 60px
!important; resize: vertical"></textarea>
                </td>
            </tr>
        </tbody>
    </table>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data() {
    return {
```

```
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
        dataSource: eventData
      },
      selectedDate: new Date(2018, 1, 15),
      group: { resources: ['Owners'] },
      ownerDataSource: [
        { text: 'Nancy', id: 1, color: '#1aaa55' },
        { text: 'Smith', id: 2, color: '#7fa900' },
        { text: 'Paul', id: 3, color: '#357cd2' }
      ],
      editorTemplate: function(e) {
        return {
          template: editorTemplateVue
        };
      }
    };
  },
  methods: {
    onPopupOpen: function(args) {
      if (args.type === 'Editor') {
        let startElement = args.element.querySelector('#StartTime');
        if (!startElement.classList.contains('e-datetimepicker')) {
          new DateTimePicker(
            { value: new Date(startElement.value) || new Date() },
            startElement
          );
        }
        let endElement = args.element.querySelector('#EndTime');
        if (!endElement.classList.contains('e-datetimepicker')) {
          new DateTimePicker(
            { value: new Date(endElement.value) || new Date() },
            endElement
          );
        }
        let processElement = args.element.querySelector('#OwnerId');
        if (!processElement.classList.contains('e-multiselect')) {
          let multiSelectObject = new MultiSelect({
            placeholder: 'Choose a owner',
            fields: { text: 'text', value: 'id' },
            dataSource: this.ownerDataSource,
            value: [args.data.OwnerId],
          });
          multiSelectObject.appendTo(processElement);
        }
      }
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
  .custom-event-editor .e-textlabel {
    padding-right: 15px;
    text-align: right;
  }
  .custom-event-editor td {
    padding: 7px;
    padding-right: 16px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/resource-field-cs1" %}

How to add recurrence options within editor template

The following code example shows how to add recurrence options within the editor template by importing RecurrenceEditor.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
      <ejs-schedule ref='scheduleObj' :height='height' :width='width'
:views='views' :selectedDate='selectedDate' :eventSettings='eventSettings'
:editorTemplate='editorTemplate' :popupOpen='onPopupOpen'>
      </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import {
    SchedulePlugin, Day, Week, WorkWeek, Month, RecurrenceEditor
} from '@syncfusion/ej2-vue-schedule';
import { DateTimePicker } from '@syncfusion/ej2-calendars';
import { eventData } from './datasource.js';
Vue.use(SchedulePlugin);
var editorTemplateVue = Vue.component('editorTemplate', {
  template: `<table class="custom-event-editor" width="100%"
cellpadding="5">
        <tbody>
            <tr>
                <td class="e-textlabel">Summary</td>
```

```
                <td colspan="4">
                    <input id="Subject" class="e-field e-input" type="text"
value="" name="Subject" style="width: 100%" />
                </td>
            </tr>
            <tr>
            <td class="e-textlabel">From</td>
                <td colspan="4">
                    <input id="StartTime" class="e-field" type="text"
name="StartTime" />
                </td>
            </tr>
            <tr>
                <td class="e-textlabel">To</td>
                <td colspan="4">
                    <input id="EndTime" class="e-field" type="text"
name="EndTime" />
                </td>
            </tr>
            <tr>
                <td colspan="4">
                    <div id='RecurrenceEditor'></div>
                </td>
            </tr>
            <tr>
                <td class="e-textlabel">Reason</td>
                <td colspan="4">
                    <textarea id="Description" class="e-field e-input"
name="Description" rows="3" cols="50" style="width: 100%; height: 60px
!important; resize: vertical"></textarea>
                </td>
            </tr>
        </tbody>
    </table>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data() {
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
        dataSource: eventData
      },
      selectedDate: new Date(2018, 1, 15),
      editorTemplate: function(e) {
        return {
          template: editorTemplateVue
        };
      }
    };
  },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  methods: {
    onPopupOpen: function(args) {
      if (args.type === 'Editor') {
        let scheduleObj = this.$refs.scheduleObj.ej2Instances;
        let startElement = args.element.querySelector('#StartTime');
        if (!startElement.classList.contains('e-datetimepicker')) {
            new DateTimePicker({ value: new Date(startElement.value) || new
Date() }, startElement);
        }
        let endElement = args.element.querySelector('#EndTime');
        if (!endElement.classList.contains('e-datetimepicker')) {
            new DateTimePicker({ value: new Date(endElement.value) || new
Date() }, endElement);
        }
        let recurElement = args.element.querySelector('#RecurrenceEditor');
        if (!recurElement.classList.contains('e-recurrenceeditor')) {
            let recurrObject = new RecurrenceEditor({});
            recurrObject.appendTo(recurElement);
            scheduleObj.eventWindow.recurrenceEditor = recurrObject;
        }
        document.getElementById('RecurrenceEditor').style.display =
(scheduleObj.currentAction == 'EditOccurrence') ? 'none' : 'block';
      }
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
  .custom-event-editor .e-textlabel {
    padding-right: 15px;
    text-align: right;
  }
  .custom-event-editor td {
    padding: 7px;
    padding-right: 16px;
  }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/editor-recurrence-cs1" %}

Apply validations on editor template fields

In the following code example, validation has been added to the status field.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
      <ejs-schedule :height='height' :width='width' :views='views'
:selectedDate='selectedDate'
        :eventSettings='eventSettings' :editorTemplate='editorTemplate'
:popupOpen='onPopupOpen'>
      </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
vue-schedule';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { DateTimePicker } from '@syncfusion/ej2-calendars';
import { FormValidator } from '@syncfusion/ej2-inputs';
import { isNullOrUndefined } from '@syncfusion/ej2-base';
import { eventData } from './datasource.js';
Vue.use(SchedulePlugin);
var editorTemplateVue = Vue.component('editorTemplate', {
  template: `<table class="custom-event-editor" width="100%"
cellpadding="5"><tbody><tr>
  <td class="e-textlabel">Summary</td><td colspan="4"><input id="Subject"
class="e-field e-input" type="text" value="" name="Subject" style="width:
100%" /></td></tr><tr><td class="e-textlabel">Status</td><td
colspan="4"><input type="text" id="EventType" name="EventType" class="e-
field" style="width: 100%" /></td></tr><tr><td class="e-
textlabel">From</td><td colspan="4"><input id="StartTime" class="e-field"
type="text" name="StartTime" /></td></tr><tr>
  <td class="e-textlabel">To</td><td colspan="4"><input id="EndTime"
class="e-field" type="text" name="EndTime" /></td></tr><tr><td class="e-
textlabel">Reason</td><td colspan="4">
  <textarea id="Description" class="e-field e-input" name="Description"
rows="3" cols="50"                  style="width: 100%; height: 60px
!important; resize: vertical"></textarea></td></tr></tbody></table>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data() {
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        dataSource: eventData
      },
      selectedDate: new Date(2018, 1, 15),
      showQuickInfo: false,
      editorTemplate: function(e) {
        return {
          template: editorTemplateVue
        };
      }
    };
  },
  methods: {
    onPopupOpen: function(args) {
      if (args.type === 'Editor') {
          if
(!isNullOrUndefined(document.getElementById("EventType_Error"))) {
              document.getElementById("EventType_Error").style.display =
"none";
              document.getElementById("EventType_Error").style.left =
"351px";
          }
          let formElement = args.element.querySelector('.e-schedule-form');
          let statusElement = args.element.querySelector('#EventType');
          if (!statusElement.classList.contains('e-dropdownlist')) {
              let dropDownListObject = new DropDownList({
                  placeholder: 'Choose status', value: statusElement.value,
                  dataSource: ['New', 'Requested', 'Confirmed'],
                  select: function(args) {
                      if
(!isNullOrUndefined(document.getElementById("EventType_Error"))) {

document.getElementById("EventType_Error").style.display = "none";
                      }
                  }
              });
              dropDownListObject.appendTo(statusElement);
          }
          let validator = formElement.ej2_instances[0];
          validator.addRules('EventType', { required: true });
          let startElement = args.element.querySelector('#StartTime');
          if (!startElement.classList.contains('e-datetimepicker')) {
              new DateTimePicker({ value: new Date(startElement.value) ||
new Date() }, startElement);
          }
          let endElement = args.element.querySelector('#EndTime');
          if (!endElement.classList.contains('e-datetimepicker')) {
              new DateTimePicker({ value: new Date(endElement.value) || new
Date() }, endElement);
          }
      }
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month]
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
.custom-event-editor .e-textlabel {
  padding-right: 15px;
  text-align: right;
}
.custom-event-editor td {
  padding: 7px;
  padding-right: 16px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-cs2" %}

How to save the customized event editor using template

The **e-field** class is not added to each field defined within the template, so you should allow to set those field values externally by using the popupClose event.

Note: You can allow to retrieve the data only on the save and delete option. Data cannot be retrieved on the close and cancel options in the editor window.

The following code example shows how to save the customized event editor using a template by the popupClose event.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
      <ejs-schedule :height='height' :width='width' :views='views'
:selectedDate='selectedDate'
        :eventSettings='eventSettings' :editorTemplate='editorTemplate'
:popupOpen='onPopupOpen' :popupClose='onPopupClose' >
      </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
vue-schedule';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { DateTimePicker } from '@syncfusion/ej2-calendars';
import { eventData } from './datasource.js';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(SchedulePlugin);
var editorTemplateVue = Vue.component('editorTemplate', {
  template: `<table class="custom-event-editor" width="100%"
cellpadding="5"><tbody><tr>
  <td class="e-textlabel">Summary</td><td colspan="4"><input id="Subject"
class="e-input" type="text" name="Subject" style="width: 100%"
/></td></tr><tr><td class="e-textlabel">Status</td><td colspan="4"><input
type="text" id="EventType" name="EventType" style="width: 100%"
/></td></tr><tr><td class="e-textlabel">From</td><td colspan="4"><input
id="StartTime" type="text" name="StartTime" /></td></tr><tr>
  <td class="e-textlabel">To</td><td colspan="4"><input id="EndTime"
type="text" name="EndTime" /></td></tr><tr><td class="e-
textlabel">Reason</td><td colspan="4">
  <textarea id="Description" class="e-input" name="Description" rows="3"
cols="50" style="width: 100%; height: 60px !important; resize:
vertical"></textarea></td></tr></tbody></table>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data() {
    return {
      height: '550px',
      width: '100%',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
        dataSource: eventData
      },
      selectedDate: new Date(2018, 1, 15),
      showQuickInfo: false,
      editorTemplate: function(e) {
        return {
          template: editorTemplateVue
        };
      }
    };
  },
  methods: {
    onPopupOpen: function(args) {
      if (args.type === 'Editor') {
          let subjectElement = args.element.querySelector('#Subject');
            if (subjectElement) {
                subjectElement.value = args.data.Subject || "";
            }
        let statusElement = args.element.querySelector('#EventType');
        if (!statusElement.classList.contains('e-dropdownlist')) {
          let dropDownListObject = new DropDownList({
            placeholder: 'Choose status',
            value: args.data.EventType,
            dataSource: ['New', 'Requested', 'Confirmed']
          });
          dropDownListObject.appendTo(statusElement);
          statusElement.setAttribute('name', 'EventType');
        }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        let startElement = args.element.querySelector('#StartTime');
        if (!startElement.classList.contains('e-datetimepicker')) {
          new DateTimePicker(
            { value: new Date(args.data.StartTime) || new Date() },
            startElement
          );
        }
        let endElement = args.element.querySelector('#EndTime');
        if (!endElement.classList.contains('e-datetimepicker')) {
          new DateTimePicker(
            { value: new Date(args.data.EndTime) || new Date() },
            endElement
          );
        }
        let descriptionElement = args.element.querySelector('#Description');
            if (descriptionElement) {
                descriptionElement.value = args.data.Description || "";
            }
      }
    },
    onPopupClose: function(args) {
      if (args.type === 'Editor'  && !isNullOrUndefined(args.data)) {
          let subjectElement = args.element.querySelector('#Subject');
            if (subjectElement) {
                args.data.Subject = subjectElement.value;
            }
        let statusElement = args.element.querySelector('#EventType');
        if (statusElement) {
          args.data.EventType = statusElement.value;
        }
        let startElement = args.element.querySelector('#StartTime');
        if (startElement) {
            args.data.StartTime = startElement.value;
        }
        let endElement = args.element.querySelector('#EndTime');
        if (endElement) {
            args.data.EndTime  = endElement.value;
        }
        let descriptionElement = args.element.querySelector('#Description');
            if (descriptionElement) {
                args.data.Description = descriptionElement.value;
            }
      }
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month]
  }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
```

```
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
.custom-event-editor .e-textlabel {
  padding-right: 15px;
  text-align: right;
}
.custom-event-editor td {
  padding: 7px;
  padding-right: 16px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-cs3" %}

In case, if you need to prevent only specific popups on Scheduler, then you can check the condition based on the popup type. The types of the popup that can be checked within the popupClose event are as follows.

| Type | Description |

|------|-------------|

| Editor | For Detailed editor window.|

| QuickInfo | For Quick popup which opens on cell click.|

| EditEventInfo |For Quick popup which opens on event click.|

| ViewEventInfo | For Quick popup which opens on responsive mode.|

| EventContainer | For more event indicator popup.|

| RecurrenceAlert | For edit recurrence event alert popup.|

| DeleteAlert | For delete confirmation popup.|

| ValidationAlert | For validation alert popup.|

| RecurrenceValidationAlert | For recurrence validation alert popup.|

Quick popups

The quick info popups are the ones that gets opened, when a cell or appointment is single clicked on the desktop mode. On single clicking a cell, you can simply provide a subject and save it. Also, while single clicking on an event, a popup will be displayed where you can get the overview of the event information. You can also edit or delete those events through the options available in it.

By default, these popups are displayed over cells and appointments of Scheduler and to disable this action, set `false` to showQuickInfo property.

The quick popup that opens while single clicking on the cells are not applicable on mobile devices.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width' :views='views'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:showQuickInfo='showQuickInfo'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, Month, TimelineViews, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['TimelineDay', 'Day', 'Week', 'Month', 'Agenda'],
      eventSettings: {
          dataSource: scheduleData
      },
      selectedDate: new Date(2018, 1, 15),
      showQuickInfo: false,
    }
  },
  provide: {
    schedule: [TimelineViews, Day, Week, Month, Agenda]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs8" %}

Copyright © 2001 -2024 Syncfusion Inc.

How to open QuickInfo popup on multiple cell selection

By default the QuickInfo popup will open on single click of the cell. To open the quick info popup on multiple cell selection, you need to select the cells and press enter key. You can open this popup immediately after multiple cell selection by setting up true to quickInfoOnSelectionEnd property where as its default value is false.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width' :views='views'
:quickInfoOnSelectionEnd='quickInfoOnSelectionEnd'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, Month, TimelineViews, Agenda } from
'@syncfusion/ej2-vue-schedule';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['TimelineDay', 'Day', 'Week', 'Month', 'Agenda'],
      quickInfoOnSelectionEnd: true,
    }
  },
  provide: {
    schedule: [TimelineViews, Day, Week, Month, Agenda]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs9" %}

Copyright © 2001 -2024 Syncfusion Inc.

How to change the watermark text of quick popup subject

By default, Add Title text is displayed on the subject field of quick popup. To change the default watermark text, change the value of the appropriate localized word collection used in the Scheduler.

`ts

L10n.load({

'en-US': {

'schedule': {

'addTitle' : 'New Title'

}

}

});

`

Customizing quick popups

The look and feel of the built-in quick popup window, which opens when single clicked on the cells or appointments can be customized by making use of the quickInfoTemplates property of the Scheduler. There are 3 sub-options available to customize them easily,

- header - Accepts the template design that customizes the header part of the quick popup.
- content - Accepts the template design that customizes the content part of the quick popup.
- footer - Accepts the template design that customizes the footer part of the quick popup.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
      <ejs-schedule :height='height' :width='width' :views='views'
        :selectedDate='selectedDate' :eventSettings='eventSettings'
        :quickInfoTemplates='quickInfoTemplates'>
      </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import {SchedulePlugin, Day, Week, WorkWeek } from '@syncfusion/ej2-vue-
schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
var headerTemplateVue = Vue.component('headerTemplate', {
  template: `<div><div v-if= "data.elementType === 'cell' "><div class="e-
cell-header">
        <div class="e-header-icon-wrapper"><button class="e-close"
title="Close"></button></div></div></div>
        <div v-else><div class="e-event-header"><div class="e-header-icon-
wrapper"><button class="e-close"
title="CLOSE"></button></div></div></div></div>`,
```

```
  data() {
    return {
      data: {}
    };
  }
});
var contentTemplateVue = Vue.component('contentTemplate', {
  template: `<div><div v-if= "data.elementType === 'cell' "><div class="e-
cell-content e-template">
        <form class="e-schedule-form"><div><input class="subject e-field"
type="text" name="Subject" placeholder="Title"></div><div><input
class="location e-field" type="text" name="Location"
placeholder="Location"></div></form></div></div><div v-else><div class="e-
event-content e-template">
        <div class="e-subject-wrap"><div><div v-if="data.Subject !==
undefined"><div class="subject">{{data.Subject}}</div></div></div><div><div
v-if="data.Location !== undefined"><div
class="location">{{data.Location}}</div></div></div><div><div v-
if="data.Description !== undefined"><div
class="description">{{data.Description}}</div></div></div></div></div></div>
</div>`,
  data() {
    return {
      data: {}
    };
  }
});
var footerTemplateVue = Vue.component('footerTemplate', {
  template: `<div><div v-if= "data.elementType === 'cell' "><div class="e-
cell-footer">
        <button class="e-event-details" title="Extra Details">Extra
Details</button>
        <button class="e-event-create" title="Add">Add</button></div></div>
        <div v-else><div class="e-event-footer"><button class="e-event-edit"
title="Edit">Edit</button>
        <button class="e-event-delete"
title="Delete">Delete</button></div></div></div>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data() {
    return {
      height: '550px',
      width: '100%',
      eventSettings: {
        dataSource: scheduleData
      },
      selectedDate: new Date(2018, 1, 15),
      views: ['Day', 'Week', 'WorkWeek'],
      quickInfoTemplates: {
        header: function(e) {
          return {
            template: headerTemplateVue
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            };
          },
          content: function(e) {
            return {
              template: contentTemplateVue
            };
          },
          footer: function(e) {
            return {
              template: footerTemplateVue
            };
          }
        }
      };
    },
    provide: {
      schedule: [Day, Week, WorkWeek]
    }
}
</script>
<style>
    @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
    @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
    .e-textlabel {
      font-weight: bold;
      padding-right: 5px;
    }
    .custom-event-editor td {
      padding: 5px;
    }
    .e-quick-popup-wrapper .e-cell-content {
      padding: 0 10px 10px 10px;
    }
    .e-quick-popup-wrapper .e-cell-content div {
      padding-bottom: 10px;
    }
    .e-quick-popup-wrapper .e-cell-content .e-field {
      border-left-width: 0;
      border-top-width: 0;
      border-right-width: 0;
      height: 25px;
      width: 100%;
    }
    .e-quick-popup-wrapper .e-event-content {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    display: flex;
  }
  .e-quick-popup-wrapper .e-event-header {
    position: absolute;
    right: 0;
  }
  .e-quick-popup-wrapper .e-cell-footer .e-event-create,
  .e-quick-popup-wrapper .e-event-footer .e-event-edit {
    position: absolute;
    right: 0;
  }
  .e-quick-popup-wrapper .e-event-content .subject {
    padding-top: 10px;
    padding-left: 10px;
    font-weight: 500;
    font-size: 14px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/quick-info-cs1" %}

The quick popup in adaptive mode can also be customized using quickInfoTemplates using e-device class.

More events indicator and popup

When the number of appointments count that lies on a particular time range * default appointment height exceeds the default height of a cell in month view and all other timeline views, a + more text indicator will be displayed at the bottom of those cells. This indicator denotes that the cell contains few more appointments in it and clicking on that will display a popup displaying all the appointments present on that day.

To disable this option of showing popup with all hidden appointments, while clicking on the text indicator, you can do code customization within the popupOpen event.

The same indicator is displayed on all-day row in calendar views such as day, week and work week views alone, when the number of appointment count present in a cell exceeds three. Clicking on the text indicator here will not open a popup, but will allow the expand/collapse option for viewing the remaining appointments present in the all-day row.

The following code example shows how to disable the display of such popups while clicking on the more text indicator.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width' :views='views'
:selectedDate='selectedDate' :currentView='currentView'
:eventSettings='eventSettings' :popupOpen='onPopupOpen'>
        </ejs-schedule>
    </div>
  </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      currentView: 'Month',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
          dataSource: scheduleData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onPopupOpen: function(args) {
        if (args.type === 'EventContainer') {
            args.cancel = true;
        }
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs10" %}

How to customize the popup that opens on more indicator

The following code example shows you how to customize the default more indicator popup in which number of events rendered count on the day has been shown in the header.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width' :views='views'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:popupOpen='onPopupOpen'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { Internationalization } from '@syncfusion/ej2-base';
import { SchedulePlugin, Month } from '@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      views: ['Month'],
      eventSettings: {
          dataSource: scheduleData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onPopupOpen: function(args) {
        if (args.type === 'EventContainer') {
            let instance = new Internationalization();
            let date  = instance.formatDate(args.data.date, { skeleton:
'MMMEd' });
            args.element.querySelector('.e-header-date').innerText = date;
            args.element.querySelector('.e-header-day').innerText = 'Event
count: ' + args.data.event.length;
        }
    }
  },
  provide: {
    schedule: [Month]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
  /* csslint ignore:start */
  .e-schedule .e-more-popup-wrapper .e-header-date {
    font-size: 15px !important;
    max-width: 44% !important;
  }
  /* csslint ignore:end */
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs11" %}

How to prevent the display of popup when clicking on the more text indicator

It is possible to prevent the display of popup window by passing the value true to cancel option within the MoreEventsClick event.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width' :views='views'
:selectedDate='selectedDate' :currentView='currentView'
:eventSettings='eventSettings' :moreEventsClick='onMoreEventsClick'>
        </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      currentView: 'Month',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
          dataSource: scheduleData
      },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  methods: {
    onMoreEventsClick: function(args) {
      args.cancel = true;
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs12" %}

How to navigate Day view when clicking on more text indicator

The following code example shows you how to customize the moreEventsClick property to navigate to the Day view when clicking on the more text indicator.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width' :views='views'
:selectedDate='selectedDate' :currentView='currentView'
:eventSettings='eventSettings' :moreEventsClick='onMoreEventsClick'>
        </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      currentView: 'Month',
      views: ['Day', 'Week', 'WorkWeek', 'Month'],
      eventSettings: {
          dataSource: scheduleData
      },
      selectedDate: new Date(2018, 1, 15),
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
  },
  methods: {
    onMoreEventsClick: function(args) {
      args.isPopupOpen = false;
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month]
  }
}
</script>
<style>
  @import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs13" %}

How to close the editor window manually

You can close the editor window by using closeEditor method.

APP.VUE

```
<template>
   <div>
      <div id='app'>
         <div id='container'>
            <tr>
               <td>
                  <div>
                     <ejs-button v-on:click.native='closeEditor'>Close
Editor Window</ejs-button>
                  </div>
               </td>
         </tr>
         <br><br>
            <ejs-schedule id='Schedule' width='100%' height='500px'
:eventSettings='eventSettings' :selectedDate='selectedDate'>
            <e-views>
               <e-view option='Day'></e-view>
```

```
                <e-view option='Week'></e-view>
                <e-view option='WorkWeek'></e-view>
                <e-view option='Month'></e-view>
            </e-views>
         </ejs-schedule>
      </div>
   </div>
</div>
</template>
<script>
    import Vue from 'vue';
    import { extend } from '@syncfusion/ej2-base';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
    import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
    Vue.use(ButtonPlugin);
    Vue.use(SchedulePlugin);
     export default {
        data () {
            return {
                eventSettings: { dataSource: extend([], scheduleData, null,
true)  },
                selectedDate: new Date(2018, 1, 15)
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        },
        methods: {
            closeEditor: function () {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                scheduleObj.closeEditor();
            },
          }
    }
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs14" %}

How to open the quick info popup manually

You can open the quick info popup in scheduler by using the openQuickInfoPopup public method. To open the cell quick info popup, you can pass the cell data as an argument to the method. To open the event quick info popup, you should pass the event data object as an argument to the method.

APP.VUE

```
<template>
  <div>
    <div id="app">
      <div id="container">
        <tr>
          <td>
            <div>
              <ejs-button v-on:click.native="onCellClick">Show Cell
QuickInfo Popup</ejs-button>
            </div>
          </td>
          <td>
            <div>
              <ejs-button v-on:click.native="onEventClick">Show Event
QuickInfo Popup</ejs-button>
            </div>
          </td>
        </tr>
        <br /><br />
        <ejs-schedule id="Schedule" width="100%" height="500px"
:eventSettings="eventSettings" :selectedDate="selectedDate">
          <e-views>
            <e-view option="Day"></e-view>
            <e-view option="Week"></e-view>
            <e-view option="WorkWeek"></e-view>
            <e-view option="Month"></e-view>
          </e-views>
        </ejs-schedule>
      </div>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { extend } from "@syncfusion/ej2-base";
import { scheduleData } from "./datasource.js";
import {
  SchedulePlugin,
  Day,
  Week,
  WorkWeek,
  Month,
} from "@syncfusion/ej2-vue-schedule";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(SchedulePlugin);
export default {
  data() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    return {
      eventSettings: {
        dataSource: [
          {
            Id: 1,
            Subject: "Review Meeting",
            StartTime: new Date(2023, 2, 5, 20, 0, 0),
            EndTime: new Date(2023, 2, 5, 21, 0, 0),
          },
        ],
      },
      selectedDate: new Date(2023, 2, 5),
    };
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month],
  },
  methods: {
    onEventClick: function () {
      let scheduleObj =
document.getElementById("Schedule").ej2_instances[0];
      let eventData = {
        Id: 1,
        Subject: "Review Meeting",
        StartTime: new Date(2023, 2, 5, 20, 0, 0),
        EndTime: new Date(2023, 2, 5, 21, 0, 0),
      };
      scheduleObj.openQuickInfoPopup(eventData, "Save");
    },
    onCellClick: function () {
      let scheduleObj =
document.getElementById("Schedule").ej2_instances[0];
      let cellData = {
        Subject: "Review Meeting",
        StartTime: new Date(2023, 2, 5, 20, 0, 0),
        EndTime: new Date(2023, 2, 5, 21, 0, 0),
      };
      scheduleObj.openQuickInfoPopup(cellData, "Add");
    },
  },
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs15" %}

How to close the quick info popup manually

You can close the quick info popup in scheduler by using the closeQuickInfoPopup public method. The following code example demonstrates the how to close quick info popup manually.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <tr>
                    <td>
                        <div>
                            <ejs-button  v-
on:click.native='closeQuickInfo'>Close QuickInfo Popup</ejs-button>
                        </div>
                    </td>
                  </tr>
                <br><br>
                <ejs-schedule id='Schedule' width='100%' height='500px'
:eventSettings='eventSettings' :selectedDate='selectedDate'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                        <e-view option='Month'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { extend } from '@syncfusion/ej2-base';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
    import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
    Vue.use(ButtonPlugin);
    Vue.use(SchedulePlugin);
     export default {
        data () {
            return {
                eventSettings: { dataSource: extend([], scheduleData, null,
true)  },
                selectedDate: new Date(2018, 1, 15)
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        },
        methods: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            closeQuickInfo: function () {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                scheduleObj.closeQuickInfoPopup();
            },
          }
    }
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/editor-window-cs16" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Recurrence editor in Vue Schedule component

The Recurrence editor is integrated into Scheduler editor window by default, to process the recurrence rule generation for events. Apart from this, it can also be used as an individual component referring from the Scheduler repository to work with the recurrence related processes.

All the valid recurrence rule string mentioned in the iCalendar specifications are applicable to use with the recurrence editor.

Customizing the repeat type option in editor

By default, there are 5 types of repeat options available in recurrence editor such as,

- Never
- Daily
- Weekly
- Monthly
- Yearly

It is possible to customize the recurrence editor to display only the specific repeat options such as Daily and Weekly options alone by setting the appropriate frequencies option.

APP.VUE

```
<template>
```

```
  <div id='app'>
    <div id='container'>
      <ejs-schedule height='550px' width='100%' :selectedDate='selectedDate'
      :eventSettings='eventSettings' :popupOpen='onPopupOpen'>
      </ejs-schedule>
    </div>
  </div>
</template>
<script>
  import Vue from 'vue';
  import { scheduleData } from './datasource.js';
  import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
  Vue.use(SchedulePlugin);
  export default {
    data () {
      return {
        selectedDate: new Date(2018, 1, 15),
        eventSettings: { dataSource: scheduleData }
      }
    },
    methods: {
        onPopupOpen: function(args) {
            if (args.type == 'Editor') {
                let recurrenceEditorObj =
                    args.element.querySelector('.e-
recurrenceeditor').ej2_instances[0];
                recurrenceEditorObj.frequencies = ['none', 'daily',
'weekly'];
            }
        }
    },
    provide: {
      schedule: [ Day, Week, WorkWeek, Month, Agenda]
    }
  }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/recurrence-cs1" %}

The other properties available in recurrence editor are tabulated below,

| Properties | Type | Description |

|------------|------|-------------|

| firstDayOfWeek | number | Sets the first day of the week on recurrence editor.|

| startDate | Date | Sets the start date from which date the recurrence event starts. |

| dateFormat | string | Sets the specific date format on recurrence editor.|

| locale | string | Sets the locale to be applied on recurrence editor.|

| cssClass | string | Allows styling to be applied on recurrence editor with custom class names.|

| enableRtl | boolean | Allows recurrence editor to render in RTL mode.|

| minDate | Date | Sets the minimum date on recurrence editor.|

| maxDate | Date | Sets the maximum date on recurrence editor.|

| value | string | Sets the recurrence rule value on recurrence editor. |

| selectedType | number | Sets the specific repeat type on the recurrence editor.|

Customizing the End Type Option in Editor

By default, there are 3 types of end options available in the recurrence editor such as:

- Never
- Until
- Count

It is possible to customize the recurrence editor to display only the specific end options, such as the `Until` and `Count` options alone, by setting the appropriate `endTypes` option.

APP.VUE

```
<template>
  <div id="app">
    <div id="container">
      <ejs-recurrenceeditor id="editor" ref="EditorObj"></ejs-
recurrenceeditor>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { RecurrenceEditorPlugin } from "@syncfusion/ej2-vue-schedule";
Vue.use(RecurrenceEditorPlugin);
export default {
  data() {
    return {};
  },
  mounted: function () {
    let recObject = this.$refs.EditorObj;
    recObject.endTypes = ["until", "count"];
  }
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-calendars/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/recurrence-cs2" %}

Accessing the recurrence rule string

The recurrence rule is usually generated based on the options selected from the recurrence editor and also it follows the iCalendar specifications. The generated recurrence rule string is a valid one to be used with the Scheduler event's recurrence rule field.

There is a change event available in recurrence editor, that triggers on every time the fields of recurrence editor tends to change. Within this event argument, you can access the generated recurrence value through the value option as shown in the following code example.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <div style='padding-bottom:15px;'>
            <label>Rule Output</label>
                <div class='rule-output-container'>
                    <div id='rule-output'></div>
                </div>
            </div>
        <ejs-recurrenceeditor :selectedType='selectedType'
:change='onChange'></ejs-recurrenceeditor>
    </div>
  </div>
</template>
<script>
  import Vue from 'vue';
  import { RecurrenceEditorPlugin } from '@syncfusion/ej2-vue-schedule';
  Vue.use(RecurrenceEditorPlugin);
  export default {
    data () {
      return {
        selectedType: 1
      }
    },
    mounted: function () {
        let outputElement = document.querySelector('#rule-output');
        outputElement.innerText = 'Select Rule';
    },
    methods: {
        onChange: function(args) {
            let outputElement = document.querySelector('#rule-output');
            if(args.value == "") {
                outputElement.innerText = 'Select Rule';
            } else {
```

```
                outputElement.innerText = args.value;
            }
        }
    }
  }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
.recurrence-editor-wrap {
        margin: 0 25%;
    }
    .rule-output-container {
        height: auto;
        border: 1px solid #969696;
    }
    #rule-output {
        padding: 8px 4px;
        text-align: center;
        min-height: 20px;
        overflow: hidden;
        overflow-wrap: break-word;
    }
    @media (max-width: 580px) {
        .recurrence-editor-wrap {
            margin: 0 5%;
        }
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/recur-editor-cs1" %}

Set specific value on recurrence editor

It is possible to display the recurrence editor with specific options loaded initially, based on the rule string that we provide. The fields of recurrence editor will change its values accordingly, when we provide a particular rule through the `setRecurrenceRule` method.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <div style='padding-bottom:15px;'>
            <label>Rule Output</label>
                <div class='rule-output-container'>
                    <div id='rule-output'></div>
                </div>
            </div>
```

```
        <ejs-recurrenceeditor id='editor' ref='EditorObj'
:selectedType='selectedType' :change='onChange'></ejs-recurrenceeditor>
    </div>
  </div>
</template>
<script>
  import Vue from 'vue';
  import { RecurrenceEditorPlugin } from '@syncfusion/ej2-vue-schedule';
  Vue.use(RecurrenceEditorPlugin);
  export default {
    data () {
      return {
        selectedType: 1
      }
    },
    mounted: function () {
        let recObject = this.$refs.EditorObj;
        recObject.setRecurrenceRule('FREQ=DAILY;INTERVAL=2;COUNT=8');
    },
    methods: {
        onChange: function(args) {
            let outputElement = document.querySelector('#rule-output');
            outputElement.innerText = args.value;
        }
    }
  }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
.recurrence-editor-wrap {
        margin: 0 25%;
    }
    .rule-output-container {
        height: auto;
        border: 1px solid #969696;
    }
    #rule-output {
        padding: 8px 4px;
        text-align: center;
        min-height: 20px;
        overflow: hidden;
        overflow-wrap: break-word;
    }
    @media (max-width: 580px) {
        .recurrence-editor-wrap {
            margin: 0 5%;
        }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/recur-editor-cs2" %}

Recurrence date generation

You can parse the recurrenceRule of an event to generate the date instances on which that particular event is going to occur, using the getRecurrenceDates method. It generates the dates based on the recurrenceRule that we provide. The parameters to be provided for getRecurrenceDates method are as follows.

| Field name | Type | Description |

|---|---|---|

| startDate | Date| Appointment start date. |

| rule | String| Recurrence rule present in an event object. |

| excludeDate | String | Date collection (in ISO format) to be excluded. It is **optional**. |

| maximumCount | Number | Number of date count to be generated. It is **optional**. |

| viewDate | Date | Current view range's first date. It is **optional**. |

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <div style='padding-bottom:15px;'>
            <label id='rule-label'>Rule Output</label>
                <div class='rule-output-container'>
                    <div id='rule-output'></div>
                </div>
            </div>
        <ejs-recurrenceeditor id='editor' style="display: none;"
ref='EditorObj' :selectedType='selectedType'>
        </ejs-recurrenceeditor>
    </div>
  </div>
</template>
<script>
  import Vue from 'vue';
  import { RecurrenceEditorPlugin } from '@syncfusion/ej2-vue-schedule';
  import { createElement } from '@syncfusion/ej2-base';
  Vue.use(RecurrenceEditorPlugin);
  export default {
    data () {
      return {
        selectedType: 1
      }
    },
    mounted: function () {
        let recObject = this.$refs.EditorObj;
        let outputElement = document.querySelector('#rule-output');
        let labelElement = document.querySelector('#rule-label');
```

```
        let dates =
recObject.getRecurrenceDates(new Date(2018, 0, 7, 10, 0),
'FREQ=DAILY;INTERVAL=1', '20180108T114224Z,20180110T114224Z', 4,
new Date(2018, 0, 7));
        labelElement.innerText = 'Date Collections';
        outputElement.innerHTML = '';
        for (let index = 0; index < dates.length; index++) {
            outputElement.appendChild(createElement('div', { innerHTML:
new Date(dates[index]).toString() }));
        }
    }
  }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
.recurrence-editor-wrap {
    margin: 0 25%;
}
.rule-output-container {
    height: auto;
    border: 1px solid #969696;
}
#rule-output {
    padding: 8px 4px;
    text-align: center;
    min-height: 20px;
    overflow: hidden;
    overflow-wrap: break-word;
}
@media (max-width: 580px) {
    .recurrence-editor-wrap {
        margin: 0 5%;
    }
}
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/recur-editor-cs3" %}

Above example will generate two dates January 7, 2018 & January 9 2018 by excluding the in between dates January 8 2018 & January 10 2018, since those dates were given in the exclusion list. Generated dates can then be utilised to create appointments.

Recurrence date generation in server-side

It is also possible to generate recurrence date instances from server-side by manually referring the `RecurrenceHelper` class, which is specifically written and referred from application end to handle this date generation process.

Copyright © 2001 -2024 Syncfusion Inc.

Refer here for the step by step procedure to achieve date generation in server-side.

Restrict date generation with specific count

In case, if the rule is given in "NEVER ENDS" category, then you can mention the maximum count when you actually want to stop the date generation starting from the provided start date. To do so, provide the appropriate `maximumCount` value within the `getRecurrenceDates` method as shown in the following code example.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <div style='padding-bottom:15px;'>
            <label id='rule-label'>Rule Output</label>
                <div class='rule-output-container'>
                    <div id='rule-output'></div>
                </div>
            </div>
        <ejs-recurrenceeditor id='editor' style="display: none;"
ref='EditorObj' :selectedType='selectedType'>
        </ejs-recurrenceeditor>
    </div>
  </div>
</template>
<script>
  import Vue from 'vue';
  import { RecurrenceEditorPlugin } from '@syncfusion/ej2-vue-schedule';
  import { createElement } from '@syncfusion/ej2-base';
  Vue.use(RecurrenceEditorPlugin);
  export default {
    data () {
      return {
        selectedType: 1
      }
    },
    mounted: function () {
        let recObject = this.$refs.EditorObj;
        let outputElement = document.querySelector('#rule-output');
        let labelElement = document.querySelector('#rule-label');
        let dates = recObject.getRecurrenceDates(new Date(2018, 0, 7, 10,
0), 'FREQ=DAILY;INTERVAL=1; COUNT=30', '20180108T114224Z,20180110T114224Z',
10, new Date(2018, 0, 7));
        labelElement.innerText = 'Date Collections';
        outputElement.innerHTML = '';
        for (let index = 0; index < dates.length; index++) {
              outputElement.appendChild(createElement('div', { innerHTML:
new Date(dates[index]).toString() }));
        }
    }
  }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
.recurrence-editor-wrap {
    margin: 0 25%;
}
.rule-output-container {
    height: auto;
    border: 1px solid #969696;
}
#rule-output {
    padding: 8px 4px;
    text-align: center;
    min-height: 20px;
    overflow: hidden;
    overflow-wrap: break-word;
}
@media (max-width: 580px) {
    .recurrence-editor-wrap {
        margin: 0 5%;
    }
}
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/recur-editor-cs4" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Timezone in Vue Schedule component

The Scheduler makes use of the current system time zone by default. If it needs to follow some other user-specific time zone, then the timezone property needs to be used. Apart from the default action of applying specific timezone to the Scheduler, it is also possible to set different time zone values for each appointments through the properties `startTimezone` and `endTimezone` which can be defined as separate fields within the event fields collection.

Note: **timezone** property only applicable for the appointment processing and current time indication.

Understanding date manipulation in JavaScript

The `new Date()` in JavaScript returns the exact current date object with complete time and timezone information. For example, it may return value such as `Wed Dec 12 2018 05:23:27 GMT+0530 (India Standard Time)` which indicates that the current date is December 12, 2018 and the current time is 5.23 AM on browsers following the IST timezone.

Scheduler with no timezone

When no specific time zone is set to Scheduler, appointments will be displayed based on the client system's timezone which is the default behavior. Here, the same appointment when viewed from different timezone will have different start and end times.

Copyright © 2001 -2024 Syncfusion Inc.

The following code example displays an appointment from 9.00 AM to 10.00 AM when you open the Scheduler from any of the timezone. This is because, we are providing the start and end time enclosing with `new Date()` which works based on the client browser's timezone.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings'>
            <e-views>
                <e-view option='Day'></e-view>
                <e-view option='Week'></e-view>
                <e-view option='TimelineWeek'></e-view>
                <e-view option='Month'></e-view>
                <e-view option='Agenda'></e-view>
            </e-views>
        </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, TimelineViews, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
Vue.use(SchedulePlugin);
let data = [{
    Id: 1,
    Subject: 'Paris',
    StartTime: new Date(2018, 1, 15, 9, 0),
    EndTime: new Date(2018, 1, 15, 10, 0)
}];
export default {
  data (){
    return {
        height: '550px',
        selectedDate: new Date(2018, 1, 15),
        eventSettings: {
            dataSource: data
        }
    }
  },
  provide: {
    schedule: [Day, Week, TimelineViews, Month, Agenda]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/time-zone-cs1" %}

Scheduler set to specific timezone

When a time zone is set to Scheduler through timezone property, the appointments will be displayed exactly based on the Scheduler timezone regardless of its client timezone. In the following code example, appointments will be displayed based on Eastern Time (UTC -05:00).

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule height='550px' :selectedDate='selectedDate'
:timezone='timezone' :eventSettings='eventSettings'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, Month } from '@syncfusion/ej2-vue-
schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 17),
                timezone: 'America/New_York'
            }
        },
        provide: {
            schedule: [Day, Week, Month]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
```

```
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/time-zone-cs2" %}

Display events on same time everywhere with no time difference

Setting timezone to UTC for Scheduler will display the appointments on same time as in the database for all the users in different time zone.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule height='550px' :selectedDate='selectedDate'
:timezone='timezone' :eventSettings='eventSettings'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { fifaEventsData } from './datasource.js';
    import { extend } from '@syncfusion/ej2-base';
    import { SchedulePlugin, Day, Week, Month } from '@syncfusion/ej2-vue-
schedule';
    import { Timezone } from '@syncfusion/ej2-schedule';
    Vue.use(SchedulePlugin);
    let timezone = new Timezone();
    var data = extend([], fifaEventsData, null, true);
    for (var i = 0; i < data.length; i++) {
        data[i].StartTime = timezone.removeLocalOffset(data[i].StartTime);
        data[i].EndTime = timezone.removeLocalOffset(data[i].EndTime);
    }
    export default {
        data () {
```

```
            return {
                eventSettings: { dataSource: data },
                selectedDate: new Date(2018, 5, 17),
                timezone: 'UTC'
            }
        },
        provide: {
            schedule: [Day, Week, Month]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/time-zone-cs3" %}

Set specific timezone for events

It is possible to set different timezone for Scheduler events by setting `startTimezone` and `endTimezone` properties within the eventSettings option. It allows each appointment to maintain different timezone and displays on Scheduler with appropriate time differences.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings'>
            <e-views>
                <e-view option='Day'></e-view>
                <e-view option='Week'></e-view>
                <e-view option='TimelineWeek'></e-view>
                <e-view option='Month'></e-view>
                <e-view option='Agenda'></e-view>
            </e-views>
        </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { SchedulePlugin, Day, Week, TimelineViews, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
Vue.use(SchedulePlugin);
let data = [{
    Id: 1,
    Subject: 'Paris',
    StartTime: new Date(2018, 1, 15, 10, 0),
    EndTime: new Date(2018, 1, 15, 12, 30),
    StartTimezone: 'Europe/Moscow',
    EndTimezone: 'Europe/Moscow'
}];
export default {
  data (){
        return {
            height: '550px',
            selectedDate: new Date(2018, 1, 15),
            eventSettings: {
                dataSource: data
            }
        }
    },
    provide: {
        schedule: [Day, Week, TimelineViews, Month, Agenda]
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/time-zone-cs4" %}

Add or remove timezone names to/from the timezone collection

Instead of displaying all the timezone names within the timezone collection (more than 200 are displayed on the editor window timezone fields by default), you can customize the timezone collection at application end as shown in the following example.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <ejs-schedule :height='height' :selectedDate='selectedDate'
:eventSettings='eventSettings'>
            <e-views>
                <e-view option='Day'></e-view>
                <e-view option='Week'></e-view>
                <e-view option='Month'></e-view>
            </e-views>
        </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, Month, timezoneData } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
let data: { [key: string]: Object }[] = [
    { Value: 'America/New_York', Text: '(UTC-05:00) Eastern Time' },
    { Value: 'UTC', Text: 'UTC' },
    { Value: 'Asia/Kolkata', Text: '(UTC+05:30) India Standard Time' }
];
timezoneData.splice(0, timezoneData.length, ...data);
export default {
    data (){
        return {
            height: '550px',
            selectedDate: new Date(2018, 1, 1),
            eventSettings: {
                dataSource: scheduleData
            }
        }
    },
    provide: {
        schedule: [Day, Week, Month]
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/time-zone-cs5" %}

Timezone methods

offset

This method is used to calculate the difference between passed UTC date and timezone.

| Parameters | Type | Description |

|------------|------|-------------|

| Date | Date | UTC time as date object.|

| Timezone | String | Timezone.|

Returns number

`

// Assume your local timezone as IST/UTC+05:30

let timezone = new Timezone();

let date: Date = new Date(2018,11,5,15,25,11);

let timeZoneOffset = timezone.offset(date,"Europe/Paris");

console.log(timeZoneOffset); //-60

`

convert

This method is used to convert the passed date from one timezone to another timezone.

| Parameters | Type | Description |

|------------|------|-------------|

| Date | Date | UTC time as date object.|

| fromOffset | number/string | Timezone from which date need to be converted.|

| toOffset | number/string | Timezone to which date need to be converted.|

Returns Date

`

// Assume your local timezone as IST/UTC+05:30

let timezone = new Timezone();

let date = new Date(2018,11,5,15,25,11);

let convertedDate = timezone.convert(date, "Europe/Paris", "Asia/Tokya");

let convertedDate1 = timezone.convert(date, 60, -360);

console.log(convertedDate); //2018-12-05T08:55:11.000Z

console.log(convertedDate1); //2018-12-05T16:55:11.000Z

`

Copyright © 2001 -2024 Syncfusion Inc.

add

This method is used to add the time difference between passed UTC date and timezone.

| Parameters | Type | Description |

|------------|------|-------------|

| Date | Date | UTC time as date object.|

| Timezone | String | Timezone.|

Returns Date

`

// Assume your local timezone as IST/UTC+05:30

let timezone = new Timezone();

let date = new Date(2018,11,5,15,25,11);

let convertedDate = timezone.add(date, "Europe/Paris");

console.log(convertedDate); //2018-12-05T05:25:11.000Z

`

remove

This method is used to remove the time difference between passed UTC date and timezone.

| Parameters | Type | Description |

|------------|------|-------------|

| Date | Date | UTC as date object.|

| Timezone | String | Timezone.|

Returns Date

`

// Assume your local timezone as IST/UTC+05:30

let timezone = new Timezone();

let date = new Date(2018,11,5,15,25,11);

let convertedDate = timezone.remove(date, "Europe/Paris");

console.log(convertedDate); //2018-12-05T14:25:11.000Z

`

removeLocalOffset

This method is used to remove the local offset time from the date passed.

| Parameters | Type | Description |

|------------|------|-------------|

| Date | Date | UTC as date object.|

Returns Date

Copyright © 2001 -2024 Syncfusion Inc.

```
// Assume your local timezone as IST/UTC+05:30
let timezone = new Timezone();
let date = new Date(2018,11,5,15,25,11);
let convertedDate = timezone.removeLocalOffset(date);
console.log(convertedDate); //2018-12-05T15:25:11.000Z
```

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Views in Vue Schedule component

The Scheduler includes wide variety of view modes with unique configuration options for each view. The available view modes are Day, Week, Work Week, Month, Year, Agenda, Month Agenda, Timeline Day, Timeline Week, Timeline Work Week and Timeline Month, Timeline Year, out of which the `Week` view is set as active.

To navigate between different views and dates, the navigation options are available at the Scheduler header bar. The active view option is usually highlighted by default. The date range of the active view will also be displayed at the left corner of the header bar, clicking on which will open a calendar popup for the ease of desired date selection.

By default, Scheduler displays the calendar views such as day, week, work week, month and agenda.

Setting specific view on scheduler

As the Scheduler displays `week` view by default, therefore to change the active view, set `currentView` property with the desired view name. The applicable view names that the Scheduler accepts are as follows,

- Day
- Week
- WorkWeek
- Month
- Year
- Agenda
- MonthAgenda
- TimelineDay
- TimelineWeek
- TimelineWorkWeek
- TimelineMonth
- TimelineYear

It is necessary to import and inject the appropriate view modules into the application to make use of these view modes on the Scheduler. Also, it is possible to display only the desired views on the Scheduler. To define and configure specific views, use the views property.

Copyright © 2001 -2024 Syncfusion Inc.

In the following example, the Scheduler displays 4 views namely, Week, Month, TimelineWeek and TimelineMonth. The appropriate view modules are imported and injected properly to display those views on the Scheduler.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='Week'></e-view>
                    <e-view option='TimelineWeek'></e-view>
                    <e-view option='Month'></e-view>
                    <e-view option='TimelineMonth'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Week, Month, TimelineViews, TimelineMonth }
from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Week, Month, TimelineViews, TimelineMonth]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/view-cs2" %}

To configure Scheduler with simply 2 views, but with different configurations on each view, refer the following code example. Here, the Week view displays the dates in dd-MM-yyyy format whereas the Month view hides the weekend days and also displays it in readonly mode.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='Week' dateFormat='dd-MMM-yyyy'></e-view>
                    <e-view option='Month' :showWeekend='showWeekend'
readonly=true></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Week, Month } from '@syncfusion/ej2-vue-
schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData },
                showWeekend: false
            }
        },
        provide: {
            schedule: [Week, Month]
        }
    }
</script>
<style>
  @import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs3" %}

View specific configuration

There are scenarios where each view may need to have different configurations. For such cases, you can define the applicable scheduler properties within the views Property for each view option as depicted in the following examples. The fields available to be used within each view options are as follows.

| Property | Type | Description | Applicable views |

|---|---|---|---|

| option | View | It accepts the Scheduler view name, based on which we can define its related properties. The view names can be Day, Week and so on. | All views.|

| isSelected | Boolean | It acts similar to the currentView property and defines the active view of the Scheduler.| All views. |

| dateFormat | Date | By default, Scheduler follows the date format as per the default culture assigned to it. When it is defined under specific view, only those assigned views follows this date format. | All views. |

| readonly | Boolean | When set to true, prevents the CRUD actions on the respective view under where it is defined. | All views. |

| resourceHeaderTemplate | String | The template option which is used to customize the resource header cells on the Scheduler. It gets applied only on the views, wherever it is defined.| All views. |

| dateHeaderTemplate | String | The template option which is used to customize the date header cells and is applied only on the views, wherever it is defined. | All views. |

| eventTemplate | String | The template option to customize the events background. It will get applied to the events of the view to which it is currently being defined. | All views. |

| showWeekend | Boolean | When set to false, it hides the weekend days of a week from the views on which it is defined.| All views. |

| group | GroupModel | Allows to set different resource grouping options on all available Scheduler view modes. | All views. |

| cellTemplate | String | The template option to customize the work cells of the Scheduler and is applied only on the views, on which it is defined. | Applicable on all views except Agenda view. |

| workDays | Number[] | It is used to set the working days on the Scheduler views. | Applicable on all views except Agenda view. |

| displayName | String | When a particular view is customized to display with different intervals, this property allows the user to set different display name for each of the views. | Applicable on all views except Agenda and Month Agenda. |

Copyright © 2001 -2024 Syncfusion Inc.

| interval | Number | It allows to customize the default Scheduler views with different set of days, weeks, work weeks or months on the applicable view type. | Applicable on all views except Agenda and Month Agenda. |

| startHour | String | It is used to specify the start hour, from which the Scheduler should be displayed. It accepts the time string in a short skeleton format and also, hides the time beyond the specified start time. | Applicable on Day, Week, Work Week, Timeline Day, Timeline Week and Timeline Work Week views. |

| endHour | String | It is used to specify the end hour, at which the Scheduler ends. It accepts the time string in a short skeleton format. | Applicable on Day, Week, Work Week, Timeline Day, Timeline Week, and Timeline Work Week views. |

| timeScale | TimeScaleModel | Allows to set different timescale configuration on each applicable view modes. | Applicable on Day, Week, Work Week, Timeline Day, Timeline Week, and Timeline Work Week views. |

| showWeekNumber | Boolean | When set to true, shows the week number on the respective weeks.| Applicable on Day, Week, Work Week, and Month views. |

| allowVirtualScrolling | Boolean | It is used to enable or disable the virtual scrolling functionality. | Applicable on Agenda and Timeline views. |

| headerRows | HeaderRowsModel | Allows defining the custom header rows on timeline views of the Scheduler to display the year, month, week, date and hour label as an individual row. | Applicable only on all timeline views. |

Day view

Usually a day view displays a single day with all its related appointments. It is possible to customize the day view to display more number of days by extending the views property with interval option. You can also define any of the above defined properties within the views object definition as depicted in the following code example.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:currentView='currentView'>
                <e-views>
                    <e-view option='Day' startHour='09:30' endHour='18:00'
:timeScale='timeScale'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData },
                currentView: 'Day',
                timeScale: {enable: true, slotCount: 5}
            }
        },
        provide: {
            schedule: [Day]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs4" %}

All the above defined properties can be accessed within Day view except `allowVirtualScrolling` and `headerRows`.

Week view

The Week view displays a count of 7 days (from Sunday to Saturday) with all its related appointments. The first day of the week can be changed using the `firstDayOfWeek` which accepts the integer (Sunday=0, Monday=1, Tuesday=2 and so on) value. You can navigate to a particular date in day view from the week view by clicking on the appropriate dates on the date header bar.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='Day' interval=2 displayName='2 Days'
startHour='09:30'
                    endHour='18:00' timeScale='timeScale'></e-view>
                    <e-view option='Week' displayName='2 Weeks' interval=2
:showWeekend='showWeekend' isSelected= true></e-view>
                </e-views>
```

```
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week } from '@syncfusion/ej2-vue-
schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData },
                timeScale: {enable: true, slotCount: 5},
                showWeekend: false
            }
        },
        provide: {
            schedule: [Day, Week]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs5" %}

All the above defined properties in the table can be accessed within Week and Work week views except `allowVirtualScrolling` and `headerRows`.

Work Week view

The Work week view displays only the working days of a week (count of 5 days) and its associated appointments. It is possible to customize the working days on the work week view by using the `workDays` property which accepts an array of integer values (such as Sunday=0, Monday=1, Tuesday=2 and so on). By default, it displays from Monday to Friday (5 days). You can also navigate to a particular date in the day view from the work week view by clicking on the appropriate dates in the date header bar.

Copyright © 2001 -2024 Syncfusion Inc.

The following code example depicts how to change the working days only on the Work Week view of the Scheduler.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='WorkWeek' :workDays='workDays'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, WorkWeek } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData },
                workDays: [2,3,5]
            }
        },
        provide: {
            schedule: [WorkWeek]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs6" %}

Copyright © 2001 -2024 Syncfusion Inc.

The Week, Work week and Day views can display the all-day row appointments in a separate all-day row with an expand/collapse option to view it.

Month view

A Month view displays the entire days of a particular month and all its related appointments. You can navigate to a particular date in the day view by clicking on the appropriate date text on the month cells.

By default, when you try to create an appointment through Month view, it is considered as created for an entire day. You can explicitly change this behavior by unchecking the All-day option from editor window, so that it defaults to the start time duration as 9.00 AM and end time as 9.30 AM.

You can also have the + more text indicator on each day cell of a Month view, clicking on which will allows you to view the hidden appointments of a day.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='Month' showWeekNumber=true readonly=true
></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Month } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Month]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs7" %}

Year view

A Year view displays all the days of a particular year with months and all its related appointments. You can navigate to a particular date in the day view by clicking on the appropriate date text on the year cells.

Year view is available in both the Horizontal and Vertical orientations. You can manage the orientation of year view through views property.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='Year' showWeekNumber=true readonly=true
></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Year } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Year]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs8" %}

The year view also has module support. In that, you can get all the months of a particular year in a calendar view format. In that calendar view, appointment contained dates are highlighted with dots placed under the individual date. When you click on the date, the event popup will be displayed and the events will be listed.

Agenda view

The Agenda view lists out the appointments in a grid-like view for the next 7 days by default from the current date. The count of the days can be changed using the API `agendaDaysCount`. It allows virtual scrolling of dates by enabling the `allowVirtualScrolling` property. Also, you can enable or disable the display of days on Scheduler that has no appointments by setting true or false to the `hideEmptyAgendaDays` property.

The following code example depicts how to customize the display of events within Agenda view alone.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='350px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='Agenda' :eventTemplate='eventTemplate'
allowVirtualScrolling=true></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { Internationalization } from '@syncfusion/ej2-base';
    import { SchedulePlugin, Agenda } from '@syncfusion/ej2-vue-schedule';
    import { eventData } from './datasource.js';
    Vue.use(SchedulePlugin);
    var instance = new Internationalization();
    var agendaTemplateVue = Vue.component('agendaTemplate', {
        template: `<div class="template-wrap"><div
class="subject">{{data.Subject}}</div><div
class="time">{{getTimeString(data.StartTime)}} -
{{getTimeString(data.EndTime)}}</div></div>`,
```

```
        data() {
            return {
                data: {}
            };
        },
        methods: {
            getTimeString: function(value) {
                return instance.formatDate(value, { skeleton: "hm"});
            }
        }
    });
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventTemplate: function(e){
                    return {
                        template: agendaTemplateVue
                    }
                },
                eventSettings: { dataSource: eventData}
            }
        },
        provide: {
            schedule: [Agenda]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
> Schedule Height is mandatory to set in pixels for Agenda view alone.
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs9" %}

Month Agenda view

A Month-Agenda view shows a month calendar, where clicking on a particular day will display the appointments present on that date below the calendar. The day with appointments are differentiated with a circular dot below the date of the calendar.

Copyright © 2001 -2024 Syncfusion Inc.

The following code example shows how to hide the weekend days on MonthAgenda view as well as the working days list is modified on Month Agenda view alone.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='MonthAgenda' :showWeekend='showWeekend'
:workDays='workDays'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SchedulePlugin, MonthAgenda } from '@syncfusion/ej2-vue-
schedule';
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData},
                workDays: [1,2,3],
                showWeekend: false
            }
        },
        provide: {
            schedule: [MonthAgenda]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/view-cs10" %}

Timeline views – Day, Week, Work Week

Similar to the day view, timeline day view shows a single day with all its appointments where the time slots are displayed horizontally. By default, the cell height adjusts as per the height set to Scheduler. When the number of appointments exceeds the visible area of the cells, the + more text indicator will be displayed at the bottom to denote the presence of few more appointments in that time range.

To make use of the timeline views (Timeline Day, Timeline Week and Timeline Work Week) on Scheduler, import and inject the module TimelineViews from the ej2-schedule package.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='TimelineDay' startHour='10:00'
endHour='15:30'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SchedulePlugin, TimelineViews } from '@syncfusion/ej2-vue-
schedule';
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData}
            }
        },
        provide: {
            schedule: [TimelineViews]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs11" %}

Similar to the Week view, the timeline week view shows 7 days with its associated appointments with the time slots displayed horizontally.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='TimelineWeek' timeScale='timeScale'></e-
view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SchedulePlugin, TimelineViews } from '@syncfusion/ej2-vue-
schedule';
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData},
                timeScale: {enable: false}
            }
        },
        provide: {
            schedule: [TimelineViews]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs12" %}

The following code example depicts how to display the timeline work week view on Scheduler,

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='TimelineWorkWeek' interval=3
:workDays='workDays' dateFormat='dd-MMM-yyyy'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SchedulePlugin, TimelineViews } from '@syncfusion/ej2-vue-
schedule';
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData},
                workDays: [1,3,5]
            }
        },
        provide: {
            schedule: [TimelineViews]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs13" %}

Clicking on the dates in the date header bar of Timeline day, Timeline week and Timeline work week will allow you to navigate to the Agenda view.

Timeline Month view

A Timeline Month view displays the current month days along with its appointments. To make use of the timeline Month view on Scheduler, import and inject TimelineMonth module from the ej2-schedule package.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='TimelineMonth'
:showWeekend='showWeekend'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SchedulePlugin, TimelineMonth } from '@syncfusion/ej2-vue-
schedule';
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData},
                showWeekend: false
            }
        },
        provide: {
            schedule: [TimelineMonth]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs14" %}

Clicking on the dates in the date header bar of Timeline month will allow you to navigate to the Timeline day view.

Timeline Year view

In Timeline Year view, each row depicts a single resource. Whereas in the vertical view, each resource is grouped parallelly as columns. Here, the resource grouping follows the tree-view like hierarchical grouping structure and can contain any level of child resources.

To make use of the timeline Year view on Scheduler, import and inject `TimelineYear` module from the `ej2-schedule` package.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                <e-views>
                    <e-view option='TimelineYear' displayName='Horizontal
Timeline Year'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SchedulePlugin, TimelineYear } from '@syncfusion/ej2-vue-
schedule';
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData},
                showWeekend: false
            }
        },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        provide: {
            schedule: [TimelineYear]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs15" %}

Resource grouping

The following code example depicts how to group the multiple resources on Timeline Year view and its relevant events are displayed accordingly under those resources.

`

<template>

<ejs-schedule id='Schedule' width='100%' height='550px' :eventSettings='eventSettings' :selectedDate='selectedDate' :currentView='currentView'>

<e-views>

<e-view option='TimelineYear' displayName='Horizontal Timeline Year' isSelected: true ></e-view>

<e-view option='TimelineYear' displayName='Vertical Timeline Year' orientation='Vertical' ></e-view>

</e-views>

<e-resources>

<e-resource field='OwnerId' title='Owner' name='Owners' :allowMultiple='allowMultiple' :dataSource='resourceDataSource' textField='OwnerText' idField='Id' colorField='OwnerColor'>

</e-resource>

</e-resources>

</ejs-schedule>

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { resourceData } from './datasource.js';
import { SchedulePlugin, TimelineYear } from '@syncfusion/ej2-vue-schedule';
Vue.use(SchedulePlugin);
export default {
data () {
return {
width: '100%',
height: '550px',
currentView: 'Week',
views: ['TimelineYear'],
selectedDate: new Date(2018, 3, 1),
allowMultiple: true,
resourceDataSource: [
{ OwnerText: 'Nancy', Id: 1, OwnerColor: '#ffaa00' },
{ OwnerText: 'Steven', Id: 2, OwnerColor: '#f8a398' },
{ OwnerText: 'Michael', Id: 3, OwnerColor: '#7499e1' }
],
eventSettings: { dataSource: resourceData }
}
},
provide: {
schedule: [TimelineYear]
}
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";

@import "../../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";

@import "../../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";

@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";

@import "../../node_modules/@syncfusion/ej2-vue-navigations/styles/material.css";

@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";

@import "../../node_modules/@syncfusion/ej2-vue-schedule/styles/material.css";

</style>

`

Auto row height

Timeline Year view supports Auto row height. When the feature rowAutoHeight is enabled, the row height gets auto-adjusted based on the number of overlapping events occupied in the same time range. If you disable the Auto row height, you have the + more text indicator on each day cell of a Timeline Year view, clicking on which will allow you to view the hidden appointments of a day.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :rowAutoHeight='rowAutoHeight'
:eventSettings='eventSettings'>
                <e-views>
                    <e-view option='TimelineYear' displayName='Horizontal
Timeline Year'></e-view>
                    <e-view option='TimelineYear' displayName='Vertical
Timeline Year' orientation='Vertical'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SchedulePlugin, TimelineYear } from '@syncfusion/ej2-vue-
schedule';
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                rowAutoHeight: true,
                eventSettings: { dataSource: scheduleData},
                showWeekend: false
            }
        },
        provide: {
            schedule: [TimelineYear]
```

```
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs16" %}

Extending view intervals

It is possible to customize the display of default number of days on different Scheduler view modes. For example, a day view can be extended to display 3 days by setting the `interval` option as 3 for the `Day` option within the `views` property as depicted in the following code example. In the same way, you can also display 2 weeks by setting interval 2 for the `Week` option.

You can provide the alternative display name for such customized views on the Scheduler header bar, by setting the appropriate `displayName` property.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:currentView='currentView'>
                <e-views>
                    <e-view option='Day' :interval='3' displayName='3
Days'></e-view>
                    <e-view option='Week' :interval='2' displayName='2
Weeks'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SchedulePlugin, Day, Week } from '@syncfusion/ej2-vue-
schedule';
    import { scheduleData } from './datasource.js';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData},
                currentView: 'Day'
            }
        },
        provide: {
            schedule: [Day, Week]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/view-cs17" %}

The view intervals can be extended on all the Scheduler view modes except Agenda and Month-Agenda views.

See Also

- How to restrict view navigation while clicking on dates

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Calendar mode in Vue Schedule component

The Scheduler supports the following two types of calendar mode.

- Gregorian Calendar
- Islamic Calendar

Gregorian Calendar

The Scheduler by default displays the gregorian calendar dates which is the most widely used calendar in the world. The Gregorian calendar is a solar calendar which has 12 months with 28 to 31 days each. A

Copyright © 2001 -2024 Syncfusion Inc.

year which is divisible by four is said to be a leap year in this calendar mode. A leap year usually has 366 days whereas the regular year has 365 days.

Islamic Calendar

The Islamic Calendar, also known as the Hijri or Muslim calendar, is a lunar calendar which has 12 months in a year with 354 or 355 days. Each month of this calendar denotes the birth of the new lunar cycle and therefore, each month can have 29 or 30 days depending on the visibility of the moon. Here, the odd-numbered months have 30 days and the even months have 29 days.

The current Islamic year is 1440 AH. Usually the Gregorian calendar runs from approximately 11 September 2018 to 30 August 2019 for this 1440 AH year.

The Scheduler has a property `calendarMode` which is used to switch between the gregorian and islamic calendar modes. By default, it is set to `Gregorian` and to use it with Islamic calendar dates, define the `calendarMode` of Scheduler to `Islamic`. The following example depicts, how to display the Islamic calendar dates on Scheduler.

To make use of Islamic calendar in Scheduler, import the `Calendar` and `Islamic` module from `ej2-calendars` package and also inject it using the `Calendar.Inject` method. Apart from this, it requires the following CLDR data to be loaded using loadCldr function.

- numberingSystems.json
- ca-gregorian.json
- numbers.json
- timeZoneNames.json
- ca-islamic.json

To know more information on, how to install the CLDR data, refer the Internationalization topic.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule height="550px" width="100%" :enableRtl='enableRtl'
calendarMode='Islamic'>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { loadCldr, setCulture, L10n } from '@syncfusion/ej2-base';
    import {
        SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, TimelineViews,
TimelineMonth,
        Resize, DragAndDrop, MonthAgenda
    } from "@syncfusion/ej2-vue-schedule";
    import { Calendar, Islamic } from '@syncfusion/ej2-calendars';
    import { scheduleData } from './datasource.js';
    import * as localeText from './locale.json';
    import * as numberingSystems from './numberingSystems.json';
    import * as ca_gregorian from './ca-gregorian.json';
    import * as numbers from './numbers.json';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    import * as timeZoneNames from './timeZoneNames.json';
    import * as ca_islamic from './ca-islamic.json';
    Calendar.Inject(Islamic)
    Vue.use(SchedulePlugin);
    L10n.load(localeText);
    loadCldr(numberingSystems, ca_gregorian, numbers, timeZoneNames,
ca_islamic);
    setCulture('ar');
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData },
                enableRtl: true,
                views: [
                    { option: 'Day' },
                    { option: 'TimelineDay' },
                    { option: 'Week' },
                    { option: 'TimelineWeek' },
                    { option: 'Month' },
                    { option: 'TimelineMonth' },
                    { option: 'Agenda' },
                    { option: 'MonthAgenda' }
                ],
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda, TimelineViews,
TimelineMonth,
    Resize, DragAndDrop, MonthAgenda]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/islamic-calendar-cs1" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Copyright © 2001 -2024 Syncfusion Inc.

Resources in Vue Schedule component

Resources and grouping support allows the Scheduler to be shared by multiple resources. Also, the appointments of each resources are displayed under relevant resources. Each resource in the Scheduler is arranged in a column/row wise order, with individual spacing to display all its respective appointments on a single page. It also supports the multiple levels of grouping of resources, thus enabling the categorization of resources in a hierarchical structure and shows it either in expandable groups (Timeline views) or else vertical hierarchy one after the other (Calendar views).

It is also possible to assign one or more resources to the same appointment, by allowing multiple selection of resource options available in the event editor window.

The HTML5 JavaScript Scheduler groups the resources based on different criteria. It includes grouping appointments based on resources, grouping resources based on dates, and timeline scheduling. Also, the data for resources bind with Scheduler either as a local JSON collection or URL, retrieving data from remote data services.

Resource fields

The default options available within the resources collection are as follows,

| Field name | Type | Description |

|---|---|---|

| field | String | A value that binds to the resource field of event object. |

| title | String | It holds the title of the resource field to be displayed on the event editor window. |

| name | String | A unique resource name used for differentiating various resource objects while grouping. |

| allowMultiple | Boolean | When set to true, allows multiple selection of resource names, thus creating multiple instances of same appointment for the selected resources. |

| dataSource | Object | Assigns the resource dataSource, where data can be passed either as an array of JavaScript objects, or else can create an instance of DataManager in case of processing remote data and can be assigned to the dataSource property. With the remote data assigned to dataSource, check the available adaptors to customize the data processing. |

| query | Query | Defines the external query that will be executed along with the data processing. |

| idField | String | Binds the resource ID field name from the resources dataSource. |

| expandedField | String | Binds the expandedField name from the resources dataSource. It usually holds boolean value which decide whether the resource of timeline views is in collapse or expand state on initial load. |

| textField | String | Binds the text field name from the resources dataSource. It usually holds the resource names. |

| groupIDField | String | Binds the group ID field name from the resource dataSource. It usually holds the value of resource IDs of parent level resources. |

| colorField | String | Binds the color field name from the resource dataSource. The color value mapped in this field will be applied to the events of resources. |

Copyright © 2001 -2024 Syncfusion Inc.

| startHourField | String | Binds the start hour field name from the resource dataSource. It allows to provide different work start hour for the resources. |

| endHourField | String | Binds the end hour field name from the resource dataSource. It allows to provide different work end hour for the resources. |

| workDaysField | String | Binds the work days field name from the resources dataSource. It allows to provide different working days collection for the resources. |

| cssClassField | String | Binds the custom CSS class field name from the resources dataSource. It maps the CSS class written for the specific resources and applies it to the events of those resources. |

Resource data binding

The data for resources can bind with Scheduler either as a local JSON collection or a service URL, retrieving resource data from remote data services.

Using local JSON data

The following code example depicts how to bind the local JSON data to the dataSource of resources collection.

```
`html

<template>

<div>

<div id='app'>

<div id='container'>

<ejs-schedule id='Schedule' width='100%' height='550px' :eventSettings='eventSettings' :selectedDate='selectedDate' :currentView='currentView'>

<e-views>

<e-view option='Week'></e-view>

<e-view option='Month'></e-view>

<e-view option='TimelineWeek'></e-view>

<e-view option='TimelineMonth'></e-view>

<e-view option='Agenda'></e-view>

</e-views>

<e-resources>

<e-resource field='OwnerId' title='Owner' name='Owners'  :dataSource='resourceDataSource' textField='OwnerText' idField='Id' colorField='OwnerColor'>

</e-resource>

</e-resources>

</ejs-schedule>

</div>

</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</template>
<script>
import Vue from 'vue';
import { resourceData } from './datasource.js';
import { SchedulePlugin, Week, Month, Agenda, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-vue-schedule';
Vue.use(SchedulePlugin);
export default {
data () {
return {
width: '100%',
height: '550px',
currentView: 'Week',
views: ['Week', 'Month', 'TimelineWeek','TimelineMonth', 'Agenda'],
selectedDate: new Date(2018, 3, 1),
resourceDataSource: [
{ OwnerText: 'Nancy', Id: 1, OwnerColor: '#ffaa00' },
{ OwnerText: 'Steven', Id: 2, OwnerColor: '#f8a398' },
{ OwnerText: 'Michael', Id: 3, OwnerColor: '#7499e1' }
],
eventSettings: { dataSource: resourceData }
}
},
provide: {
schedule: [Week, Month, Agenda, TimelineViews, TimelineMonth]
}
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-schedule/styles/material.css";
</style>
`
```

Using remote service URL

The following code example depicts how to bind the remote data for resources dataSource.

```
`html
<template>
<div>
<div id='app'>
<div id='container'>
<ejs-schedule id='Schedule' width='100%' height='550px' :eventSettings='eventSettings'
:selectedDate='selectedDate' :currentView='currentView'>
<e-views>
<e-view option='Week'></e-view>
<e-view option='Month'></e-view>
<e-view option='TimelineWeek'></e-view>
<e-view option='TimelineMonth'></e-view>
<e-view option='Agenda'></e-view>
</e-views>
<e-resources>
<e-resource field='OwnerId' title='Owner' name='Owners'  :dataSource='resourceDataSource'
textField='OwnerText' idField='Id' colorField='OwnerColor'>
</e-resource>
</e-resources>
</ejs-schedule>
</div>
</div>
</div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import { SchedulePlugin, Week, Month, Agenda, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-vue-schedule';
import { resourceData } from './datasource.js';
Vue.use(SchedulePlugin);
let resource = new DataManager({
url: 'Home/GetResourceData',
adaptor: new UrlAdaptor,
crossDomain: true
});
export default {
data () {
return {
width: '100%',
height: '550px',
currentView: 'Week',
views: ['Week', 'Month', 'TimelineWeek','TimelineMonth', 'Agenda'],
selectedDate: new Date(2018, 3, 1),
resourceDataSource: resource,
eventSettings: { dataSource: resourceData }
}
},
provide: {
schedule: [Week, Month, Agenda, TimelineViews, TimelineMonth]
}
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";

@import "../../node_modules/@syncfusion/ej2-vue-navigations/styles/material.css";

@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";

@import "../../node_modules/@syncfusion/ej2-vue-schedule/styles/material.css";

</style>

`

Scheduler with multiple resources

It is possible to display the Scheduler in default mode without visually showcasing all the resources in it, but allowing to assign the required resources to the appointments through the event editor resource options.

The appointments belonging to the different resources will be displayed altogether on the default Scheduler, which will be differentiated based on the resource color assigned in the **Link to the Video** (depicting to which resource that particular appointment belongs) collection.

Learn how to add appointments from multiple resources to Vue Scheduler by watching this video.

Example: To display default Scheduler with multiple resource options in the event editor, ignore the group option and simply define the resources property with all its internal options.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate'
:currentView='currentView'>
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='OwnerId' title='Owner'
name='Owners' :allowMultiple='allowMultiple'
:dataSource='resourceDataSource' textField='OwnerText' idField='Id'
colorField='OwnerColor'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { resourceData } from './datasource.js';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    import { SchedulePlugin, Week, Month, Agenda, TimelineViews,
TimelineMonth } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                width: '100%',
                height: '550px',
                currentView: 'Week',
                views: ['Week', 'Month', 'TimelineWeek','TimelineMonth',
'Agenda'],
                selectedDate: new Date(2018, 3, 1),
                allowMultiple: true,
                resourceDataSource: [
                    { OwnerText: 'Nancy', Id: 1, OwnerColor: '#ffaa00' },
                    { OwnerText: 'Steven', Id: 2, OwnerColor: '#f8a398' },
                    { OwnerText: 'Michael', Id: 3, OwnerColor: '#7499e1' }
                ],
                eventSettings: { dataSource: resourceData }
            }
        },
        provide: {
            schedule: [Week, Month, Agenda, TimelineViews, TimelineMonth]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/multiple-resources-cs1" %}

Setting `allowMultiple` to `true` in the above code example allows you to select multiple resources from the event editor and also creates multiple copies of the same appointment in the Scheduler for each resources while rendering.

Resource grouping

Resource grouping support allows the Scheduler to group the resources in a hierarchical structure both as an expandable groups (Timeline views) and as vertical hierarchy displaying resources one after the other (Resources view).

Copyright © 2001 -2024 Syncfusion Inc.

Scheduler supports both single and multiple levels of resource grouping that can be customized both in timeline and vertical Scheduler views.

Vertical resource view

The following code example displays how the multiple resources are grouped and its events are portrayed in the default calendar views.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' width='100%' height='550px'
            :eventSettings='eventSettings' :currentView='currentView'
:selectedDate='selectedDate' :group='group'>
                <e-views>
                    <e-view option='Week'></e-view>
                    <e-view option='Month'></e-view>
                    <e-view option='Agenda'></e-view>
                </e-views>
                <e-resources>
                    <e-resource field='ProjectId' title='Choose Project'
name='Projects' :dataSource='projectResourceDataSource' textField='text'
                    idField='id' colorField='color'>
                    </e-resource>
                    <e-resource field='CategoryId' title='Category'
name='Categories' :dataSource='categoryResourceDataSource'
:allowMultiple='allowMultiple'
                    textField='text' idField='id' colorField='color'>
                    </e-resource>
                </e-resources>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { resourceProjectData } from './datasource.js';
    import { SchedulePlugin, Week, Month, Agenda } from '@syncfusion/ej2-
vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                currentView: 'Week',
                selectedDate: new Date(2018, 3, 4),
                group: {
                    byGroupID: false,
                    resources: ['Projects', 'Categories']
                },
                projectResourceDataSource: [
                    { text: 'PROJECT 1', id: 1, color: '#cb6bb2' },
                    { text: 'PROJECT 2', id: 2, color: '#56ca85' },
                    { text: 'PROJECT 3', id: 3, color: '#df5286' }
                ],
                categoryResourceDataSource: [
                    { text: 'Development', id: 1, color: '#df5286' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    { text: 'Testing', id: 2, color: '#7fa900' }
                ],
                eventSettings: { dataSource: resourceProjectData },
                allowMultiple: true
            }
        },
        provide: {
            schedule: [Week, Month, Agenda]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/resource-grouping-cs1" %}

Timeline resource view

The following code example depicts how to group the multiple resources on Timeline Scheduler views and its relevant events are displayed accordingly under those resources.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate'
:currentView='currentView' :group='group'>
                    <e-views>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field= 'RoomId' title= 'Room' name=
'Rooms' :dataSource='roomDataSource' textField='RoomText' idField='Id'
colorField='RoomColor'>
                        </e-resource>
                        <e-resource field='OwnerId' title='Owner'
name='Owners' :allowMultiple='allowMultiple' :dataSource='ownerDataSource'
```

```
                        textField='OwnerText' idField='Id' groupIDField=
'OwnerGroupId' colorField='OwnerColor'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { resourceData } from './datasource.js';
    import { SchedulePlugin, TimelineViews, TimelineMonth, Agenda } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                width: '100%',
                height: '550px',
                currentView: 'TimelineWeek',
                selectedDate: new Date(2018, 3, 4),
                group: {
                    resources: ['Rooms', 'Owners']
                },
                roomDataSource: [
                    { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2' },
                    { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
                ],
                allowMultiple: true,
                ownerDataSource: [
                    { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1,
OwnerColor: '#ffaa00' },
                    { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2,
OwnerColor: '#f8a398' },
                    { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1,
OwnerColor: '#7499e1' }
                ],
                eventSettings: { dataSource: resourceData },
            }
        },
        provide: {
            schedule: [TimelineViews, TimelineMonth, Agenda]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/resource-grouping-cs2" %}

Grouping single-level resources

This kind of grouping allows the Scheduler to display all the resources at a single level simultaneously. The appointments mapped under resources will be displayed with the colors as per the colorField defined on the resources collection.

Example: To display the Scheduler with single level resource grouping,

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate' :group='group'>
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='OwnerId' title='Owner'
name='Owners' :allowMultiple='allowMultiple' :dataSource='ownerDataSource'
                        textField='OwnerText' idField='Id'
colorField='OwnerColor'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { resourceData } from './datasource.js';
    import { SchedulePlugin, Week, Month, Agenda, TimelineViews,
TimelineMonth } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                width: '100%',
                height: '550px',
                currentView: 'TimelineWeek',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                selectedDate: new Date(2018, 3, 4),
                group: {
                    resources: ['Owners']
                },
                allowMultiple: true,
                ownerDataSource: [
                    { OwnerText: 'Nancy', Id: 1, OwnerColor: '#ffaa00' },
                    { OwnerText: 'Steven', Id: 2, OwnerColor: '#f8a398' },
                    { OwnerText: 'Michael', Id: 3, OwnerColor: '#7499e1' }
                ],
                eventSettings: { dataSource: resourceData },
            }
        },
        provide: {
            schedule: [Week, Month, Agenda, TimelineViews, TimelineMonth]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/single-level-resource-cs1" %}

The `name` field defined in the **resources** collection namely `Owners` will be mapped within the `group` property, in order to enable the grouping option with those resource levels on the Scheduler.

Grouping multi-level resources

It is possible to group the resources of Scheduler in multiple levels, by mapping the child resources to each parent resource. In the following example, there are 2 levels of resources, on which the second level resources are defined with `Link to the Video` mapping to the first level resource's ID so as to establish the parent-child relationship between them.

To get start quickly about Grouping Multiple Resources in Vue Scheduler, you can check on this video:

Example: To display the Scheduler with multiple level resource grouping options,

APP.VUE

```
<template>
    <div>
        <div id='app'>
```

```
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate' :group='group'>
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='RoomId' title='Room' name='Rooms'
:dataSource='roomDataSource' textField='RoomText' idField='Id'
colorField='RoomColor'>
                        </e-resource>
                        <e-resource field='OwnerId' title='Owner'
name='Owners' :allowMultiple='allowMultiple' :dataSource='ownerDataSource'
                        textField='OwnerText' idField='Id' groupIDField=
'OwnerGroupId' colorField='OwnerColor'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { resourceData } from './datasource.js';
    import { SchedulePlugin, Week, Month, Agenda, TimelineViews,
TimelineMonth } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                width: '100%',
                height: '550px',
                selectedDate: new Date(2018, 3, 4),
                group: {
                    resources: ['Rooms', 'Owners']
                },
                roomDataSource: [
                    { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2' },
                    { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
                ],
                allowMultiple: true,
                ownerDataSource: [
                    { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1,
OwnerColor: '#ffaa00' },
                    { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2,
OwnerColor: '#f8a398' },
                    { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1,
OwnerColor: '#7499e1' }
                ],
                eventSettings: { dataSource: resourceData },
            }
        },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        provide: {
            schedule: [Week, Month, Agenda, TimelineViews, TimelineMonth]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/resource-grouping-cs3" %}

One-to-One grouping

In multi-level grouping, Scheduler usually groups the resources on the child level based on the GroupID that maps with the Id field of parent level resources (as byGroupID set to true by default). There are also option which allows you to group all the child resource(s) against each of its parent resource(s). To enable this kind of grouping, set false to the byGroupID option within the group property. In the following code example, there are two levels of resources, on which all the 3 resources at the child level is mapped one to one with each resource on the first level.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate' :group='group'>
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field= 'RoomId' title= 'Room' name=
'Rooms' :dataSource='roomDataSource' textField='RoomText' idField='Id'
colorField='RoomColor'>
                        </e-resource>
                        <e-resource field='OwnerId' title='Owner'
name='Owners' :allowMultiple='allowMultiple' :dataSource='ownerDataSource'
```

```
                        textField='OwnerText' idField='Id' groupIDField=
'OwnerGroupId' colorField='OwnerColor'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { resourceData } from './datasource.js';
    import { SchedulePlugin, Week, Month, Agenda, TimelineViews,
TimelineMonth } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                width: '100%',
                height: '550px',
                selectedDate: new Date(2018, 3, 4),
                currentView: 'Week',
                group: {
                    byGroupID: false,
                    resources: ['Rooms', 'Owners']
                },
                roomDataSource: [
                    { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2' },
                    { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
                ],
                allowMultiple: true,
                ownerDataSource: [
                    { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1,
OwnerColor: '#ffaa00' },
                    { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2,
OwnerColor: '#f8a398' },
                    { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1,
OwnerColor: '#7499e1' }
                ],
                eventSettings: { dataSource: resourceData },
            }
        },
        provide: {
            schedule: [Week, Month, Agenda, TimelineViews, TimelineMonth]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/resource-grouping-cs4" %}

Grouping resources by date

It groups the number of resources under each date and is applicable only on the calendar views such as Day, Week, Work Week, Month, Agenda and Month-Agenda. To enable such grouping, set byDate option to true within the group property.

Example: To display the Scheduler with resources grouped by date,

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate' :group='group'>
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='OwnerId' title='Owner'
name='Owners' :allowMultiple='allowMultiple' :dataSource='ownerDataSource'
                        textField='text' idField='id' colorField='color'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { resourceData } from './datasource.js';
    import { SchedulePlugin, Week, Month, Agenda} from '@syncfusion/ej2-vue-
schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                width: '100%',
                height: '550px',
                selectedDate: new Date(2018, 3, 4),
                group: {
                    byDate: true,
                    resources: ['Owners']
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                },
                allowMultiple: true,
                ownerDataSource: [
                    { text: 'Alice', id: 1, color: '#1aaa55' },
                    { text: 'Smith', id: 2, color: '#7fa900' }
                ],
                eventSettings: { dataSource: resourceData },
            }
        },
        provide: {
            schedule: [Week, Month, Agenda]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/single-level-resource-cs2" %}

This kind of grouping by date is not applicable on any of the timeline views.

Customizing parent resource cells

In timeline view work cells of parent resource can be customized by checking the `elementType` as `resourceGroupCells` in the event `renderCell`. In the following code example, background color of the work hours has been changed.

APP.VUE

```
<template>
    <div>
        <div id='container'>
            <ejs-schedule ref='scheduleObj' id='Schedule' width='100%'
height='550px' :eventSettings='eventSettings'
            :selectedDate='selectedDate' :currentView='currentView'
:group='group' :renderCell='onRenderCell'>
                <e-views>
                    <e-view option='TimelineDay'></e-view>
                    <e-view option='TimelineWeek'></e-view>
                    <e-view option='TimelineMonth'></e-view>
                </e-views>
                <e-resources>
```

```
                    <e-resource field='RoomId' title='Room' name='Rooms'
:dataSource='roomDataSource' textField='RoomText' idField='Id' groupIDField=
'RoomGroupId' colorField='RoomColor'>
                    </e-resource>
                    <e-resource field='OwnerId' title='Owner' name='Owners'
:allowMultiple='allowMultiple' :dataSource='ownerDataSource'
                    textField='OwnerText' idField='Id' groupIDField=
'OwnerGroupId' colorField='OwnerColor'>
                    </e-resource>
                </e-resources>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { resourceData } from './datasource.js';
    import { SchedulePlugin, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 3, 4),
                currentView: 'TimelineWeek',
                group: {
                    resources: ['Rooms', 'Owners']
                },
                roomDataSource: [
                    { RoomText: 'ROOM 1', Id: 1, RoomGroupId: 1, RoomColor:
'#cb6bb2' },
                    { RoomText: 'ROOM 2', Id: 2, RoomGroupId: 2, RoomColor:
'#56ca85' }
                ],
                allowMultiple: true,
                ownerDataSource: [
                    { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1,
OwnerColor: '#ffaa00' },
                    { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2,
OwnerColor: '#f8a398' },
                    { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1,
OwnerColor: '#7499e1' }
                ],
                eventSettings: { dataSource: resourceData }
            }
        },
        provide: {
            schedule: [TimelineViews, TimelineMonth]
        },
        methods: {
            onRenderCell(args) {
                if (args.elementType == 'resourceGroupCells' &&
args.element.className.indexOf('e-work-hours') > 0) {
                    args.element.style.background = "#FAFAE3";
                }
            }
        }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/resource-grouping-cs5" %}

Working with shared events

Multiple resources can share the same events, thus allowing the CRUD action made on it to reflect on all other shared instances simultaneously. To enable such option, set `allowGroupEdit` option to `true` within the `group` property. With this property enabled, a single appointment

object will be maintained within the appointment collection, even if it is shared by more than one resource – whereas the resource fields of such appointment object will be in array which hold the IDs of the multiple resources.

Any actions such as create, edit or delete held on any one of the shared event instances, will be reflected on all other related instances visible on the UI.

Example: To edit all the resource events simultaneously,

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:currentView='currentView' :group='group'>
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='ConferenceId' title='Conference'
name='Conferences' :allowMultiple='allowMultiple'
:dataSource='resourceDataSource'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                        textField='Text' idField='Id' colorField='Color'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SchedulePlugin, Week, Month, Agenda, TimelineViews,
TimelineMonth, Resize} from '@syncfusion/ej2-vue-schedule';
    import { resourceConferenceData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 5, 2),
                currentView: 'Week',
                group: {
                    allowGroupEdit: true,
                    resources: ['Conferences']
                },
                resourceDataSource: [
                    { Text: 'Margaret', Id: 1, Color: '#1aaa55' },
                    { Text: 'Robert', Id: 2, Color: '#357cd2' },
                    { Text: 'Laura', Id: 3, Color: '#7fa900' }
                ],
                allowMultiple: true,
                eventSettings: { dataSource: resourceConferenceData }
            }
        },
        provide: {
            schedule: [Week, Month, Agenda, TimelineViews, TimelineMonth,
Resize]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/resource-grouping-cs6" %}

Simple resource header customization

It is possible to customize the resource header cells using built-in template option and change the look and appearance of it in both the vertical and timeline view modes. All the resource related fields and other information can be accessed within the resource header template option.

Example: To customize the resource header and display it along with designation field, refer the below code example.

APP.VUE

```
<template>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate' :group='group'
:resourceHeaderTemplate='resourceHeaderTemplate'>
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='DoctorId' title='Doctor Name'
name='Doctors' :dataSource='ownerDataSource'
                        textField='text' idField='id' colorField='color'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
</template>
<script>
    import Vue from 'vue';
    import { doctorData} from './datasource.js';
    import { SchedulePlugin, Week, Month, Agenda, TimelineViews,
TimelineMonth} from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    var resourceHeaderTemplateVue = Vue.component('headerTemplate', {
        template: `<div class='template-wrap'><div class="resource-
details"><div class="resource-name">{{data.resourceData.text}}</div><div
class="resource-
designation">{{data.resourceData.designation}}</div></div></div>`,
        data() {
            return {
                data: {}
            };
        }
    });
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 3, 4),
                group: {
```

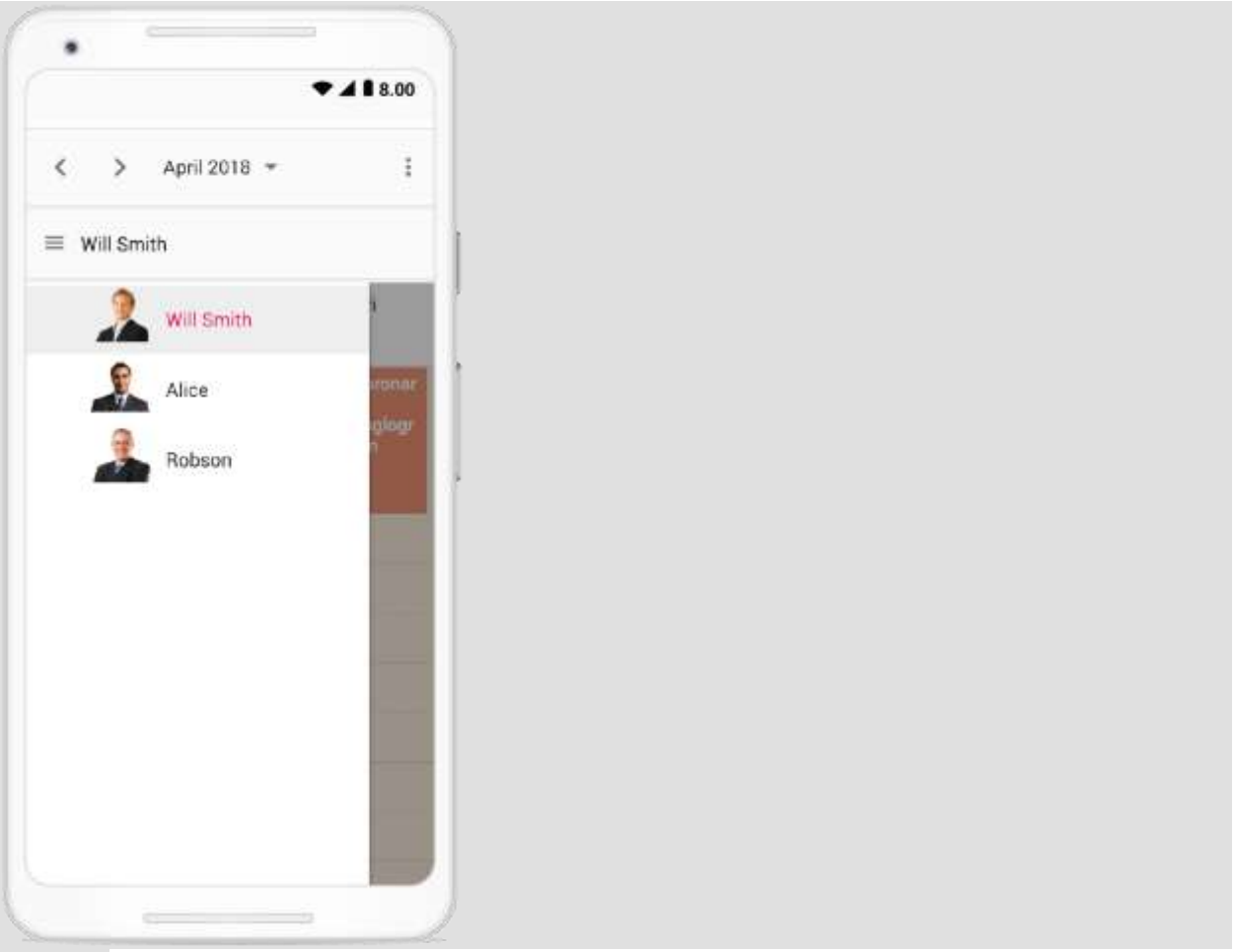
```
                    resources: ['Doctors']
                },
                ownerDataSource: [
                    { text: 'Will Smith', id: 1, color: '#ea7a57',
designation: 'Cardioligst' },
                    { text: 'Alice', id: 2, color: '#7fa900', designation:
'Neurologist' },
                    { text: 'Robson', id: 3, color: '#7fa900', designation:
'Orthopedic Surgeon'  }
                ],
                resourceHeaderTemplate: function(e){
                    return {
                        template: resourceHeaderTemplateVue
                    };
                },
                eventSettings: { dataSource: doctorData },
            }
        },
        provide: {
            schedule: [Week, Month, Agenda, TimelineViews, TimelineMonth]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
  .e-schedule .e-vertical-view .e-resource-cells {
        height: 62px;
    }
    .e-schedule .template-wrap {
        display: flex;
        text-align: left;
    }
    .e-schedule .template-wrap .resource-details {
        padding-left: 10px;
    }
    .e-schedule .template-wrap .resource-details .resource-name {
        font-size: 16px;
        font-weight: 500;
        margin-top: 5px;
    }
    .e-schedule.e-device .template-wrap .resource-details .resource-name {
        font-size: inherit;
        font-weight: inherit;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
    .e-schedule.e-device .e-resource-tree-popup .e-fullrow {
        height: 50px;
    }
    .e-schedule.e-device .template-wrap .resource-details .resource-
designation {
        display: none;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/resource-header-cs1" %}

To customize the resource header in compact mode properly make use of the class



e-device as in the code example.

![Resource header template in compact mode](./images/header-template.png)

Customizing resource header with multiple columns

It is possible to customize the resource headers to display with multiple columns such as Room, Type and Capacity. The following code example depicts the way to achieve it and is applicable only on timeline views.

APP.VUE

```
<template>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate' :group='group'
:resourceHeaderTemplate='resourceHeaderTemplate' :renderCell='onRenderCell'>
                    <e-views>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='RoomId' title='RoomType'
name='MeetingRoom'  :dataSource='ownerDataSource'
                        textField='text' idField='id' colorField='color'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
</template>
<script>
    import Vue from 'vue';
    import { roomData} from './datasource.js';
    import { SchedulePlugin, TimelineViews, TimelineMonth} from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    var resourceHeaderTemplateVue = Vue.component('headerTemplate', {
        template: `<div class='template-wrap'>
                <div class="room-name">{{data.resourceData.text}}</div>
                <div class="room-type">{{data.resourceData.type}}</div>
                <div class="room-
capacity">{{data.resourceData.capacity}}</div>
                </div>`,
        data() {
            return {
                data: {}
            };
        }
    });
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 7, 1),
                group: {
                    resources: ['MeetingRoom']
                },
                ownerDataSource: [
                    { text: 'Jammy', id: 1, color: '#ea7a57', capacity: 20,
type: 'Conference' },
                    { text: 'Tweety', id: 2, color: '#7fa900', capacity: 7,
type: 'Cabin' },
                    { text: 'Nestle', id: 3, color: '#5978ee', capacity: 5,
type: 'Cabin' },
                    { text: 'Phoenix', id: 4, color: '#fec200', capacity:
15, type: 'Conference' },
                    { text: 'Mission', id: 5, color: '#df5286', capacity:
25, type: 'Conference' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    { text: 'Hangout', id: 6, color: '#00bdae', capacity:
10, type: 'Cabin' },
                    { text: 'Rick Roll', id: 7, color: '#865fcf', capacity:
20, type: 'Conference' },
                    { text: 'Rainbow', id: 8, color: '#1aaa55', capacity: 8,
type: 'Cabin' },
                    { text: 'Swarm', id: 9, color: '#df5286', capacity: 30,
type: 'Conference' },
                    { text: 'Photogenic', id: 10, color: '#710193',
capacity: 25, type: 'Conference' }
                ],
                resourceHeaderTemplate: function(e){
                     return {
                        template: resourceHeaderTemplateVue
                    };
                },
                eventSettings: { dataSource: roomData },
            }
        },
        methods: {
            onRenderCell: function(args){
                if (args.elementType === 'emptyCells' &&
args.element.classList.contains('e-resource-left-td')) {
                    let target = args.element.querySelector('.e-resource-
text');
                    target.innerHTML = '<div class="name">Rooms</div><div
class="type">Type</div><div class="capacity">Capacity</div>';
                }
            }
        },
        provide: {
            schedule: [TimelineViews, TimelineMonth]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
  .e-schedule .e-timeline-view .e-resource-left-td {
        vertical-align: bottom;
    }
    .e-schedule .e-timeline-view .e-resource-left-td .e-resource-text {
        display: flex;
        font-weight: 500;
```

```
        padding: 0;
    }
    .e-schedule .e-timeline-view .e-resource-left-td .e-resource-text>div {
        border-right: 1px solid rgba(0, 0, 0, 0.12);
        border-top: 1px solid rgba(0, 0, 0, 0.12);
        flex: 0 0 33.3%;
        font-weight: 500;
        height: 36px;
        line-height: 34px;
        padding-left: 5px;
    }
    .e-schedule .e-timeline-view .e-resource-left-td .e-resource-
text>div:last-child {
        border-right: 0;
    }
    .e-schedule .template-wrap {
        display: flex;
        height: 100%;
        text-align: left;
    }
    .e-schedule .template-wrap>div {
        border-right: 1px solid rgba(0, 0, 0, 0.12);
        flex: 0 0 33.3%;
        font-weight: 500;
        line-height: 58px;
        overflow: hidden;
        padding-left: 5px;
        text-overflow: ellipsis;
    }
    .e-schedule .template-wrap>div:last-child {
        border-right: 0;
    }
    .e-schedule .e-timeline-view .e-resource-cells,
    .e-schedule .e-timeline-month-view .e-resource-cells {
        padding-left: 0;
    }
    .e-schedule .e-timeline-view .e-date-header-wrap table col,
    .e-schedule .e-timeline-view .e-content-wrap table col {
        width: 100px;
    }
    @media (max-width: 550px) {
        .e-schedule .e-timeline-view .e-resource-left-td {
            width: 100px;
        }
        .e-schedule .e-timeline-view .e-resource-left-td .e-resource-
text>div,
        .e-schedule .template-wrap>div {
            flex: 0 0 100%;
        }
        .e-schedule .template-wrap>div:first-child {
            border-right: 0;
        }
        .e-schedule .e-timeline-view .e-resource-left-td .e-resource-
text>div:first-child {
            border-right: 0;
        }
        .e-schedule .room-type,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        .e-schedule .room-capacity {
            display: none;
        }
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/resource-header-column-customization-cs1" %}

Collapse/Expand child resources in timeline views

It is possible to expand and collapse the resources which have child resource in timeline views dynamically. By default, resources are in expanded state with their child resource. We can collapse and expand the child resources in UI by setting `expandedField` option as `false` whereas its default value is `true`.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate'
:currentView='currentView' :group='group'>
                    <e-views>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field= 'RoomId' title= 'Room' name=
'Rooms' :dataSource='roomDataSource' textField='RoomText' idField='Id'
colorField='RoomColor' expandedField='IsExpand'>
                        </e-resource>
                        <e-resource field='OwnerId' title='Owner'
name='Owners' :allowMultiple='allowMultiple' :dataSource='ownerDataSource'
                        textField='OwnerText' idField='Id' groupIDField=
'OwnerGroupId' colorField='OwnerColor'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { resourceData } from './datasource.js';
    import { SchedulePlugin, TimelineViews, TimelineMonth, Agenda } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                width: '100%',
```

```
                height: '550px',
                currentView: 'TimelineWeek',
                selectedDate: new Date(2018, 3, 4),
                group: {
                    resources: ['Rooms', 'Owners']
                },
                roomDataSource: [
                    { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2',
IsExpand: false },
                    { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
                ],
                allowMultiple: true,
                ownerDataSource: [
                    { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1,
OwnerColor: '#ffaa00' },
                    { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2,
OwnerColor: '#f8a398' },
                    { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1,
OwnerColor: '#7499e1' }
                ],
                eventSettings: { dataSource: resourceData },
            }
        },
        provide: {
            schedule: [TimelineViews, TimelineMonth, Agenda]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/resource-grouping-cs8" %}

Displaying tooltip for resource headers

It is possible to display tooltips over the resource headers showing the resource information. By default, there won't be any tooltips displayed on the resource headers, and to enable it, you need to assign the customized template design to the `headerTooltipTemplate` option within the `group` property.

APP.VUE

```
<template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate' :group='group'>
                    <e-views>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='OwnerId' title='Owner'
name='Owners' :dataSource='ownerDataSource' :allowMultiple='allowMultiple'
                        textField='OwnerText' idField='Id'
groupIDField='OwnerGroupId' colorField='OwnerColor'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
</template>
<script>
    import Vue from 'vue';
    import { SchedulePlugin, Week, Month, TimelineViews, TimelineMonth }
from '@syncfusion/ej2-vue-schedule';
    import { resourceData } from './datasource.js';
    Vue.use(SchedulePlugin);
    var headerTooltipTemplateVue = Vue.component('headerTooltipTemplate', {
        template: `<div class='template-wrap'><div class="resource-
text">Name:{{data.resourceData.OwnerText}}</div></div>`,
        data() {
            return {
                data: {}
            };
        }
    });
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 3, 4),
                group: {
                  resources: ['Rooms', 'Owners'],
                  headerTooltipTemplate: function(e){
                      return {
                          template: headerTooltipTemplateVue
                      };
                  }
                },
                allowMultiple: true,
                ownerDataSource: [
                    { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1,
OwnerColor: '#ffaa00' },
                    { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2,
OwnerColor: '#f8a398' },
                    { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1,
OwnerColor: '#7499e1' }
                ],
                eventSettings: { dataSource: resourceData },
            }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        },
        provide: {
            schedule: [TimelineViews, TimelineMonth]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
  .e-schedule .e-vertical-view .e-resource-cells {
        height: 45px;
    }
    .e-schedule .e-agenda-view .template-wrap .resource-text {
        text-align: center;
    }
    .e-schedule .template-wrap .resource-text {
        font-size: 15px;
        padding: 4px 4px 4px;
        height: 25px;
        text-overflow: ellipsis;
        white-space: nowrap;
        overflow: hidden;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/header-tooltip-cs1" %}

Choosing among resource colors for appointments

By default, the colors defined on the top level resources collection will be applied for the events. In case, if you want to apply specific resource color to events irrespective of its top-level parent resource color, it can be achieved by defining `resourceColorField` option within the `eventSettings` property.

In the following example, the colors mentioned in the second level will get applied over the events.

APP.VUE

```
<template>
    <div>
    <div id='container' class='col-lg-12'>
        <div class='content-wrapper'>
            <div class='col-lg-12 property-section'>
                <table id='property' title='Event Trace'>
                    <tbody>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                        <tr>
                            <td>
                                <ejs-radiobutton label="Rooms"
name="default" value="Rooms" :checked='true'
                                    :change='onChange'></ejs-radiobutton>
                            </td>
                            <td>
                                <ejs-radiobutton label='Owners'
name="default" value="Owners" :checked='false'
                                    :change='onChange'></ejs-radiobutton>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
            <div>
                <ejs-schedule ref='scheduleObj' id='Schedule' width='100%'
height='500px' :eventSettings='eventSettings'
                    :selectedDate='selectedDate' :group='group'>
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='RoomId' title='Room' name='Rooms'
:dataSource='roomDataSource'
                            textField='RoomText' idField='Id'
groupIDField='RoomGroupId' colorField='RoomColor'>
                        </e-resource>
                        <e-resource field='OwnerId' title='Owner'
name='Owners' :allowMultiple='allowMultiple'
                            :dataSource='ownerDataSource'
textField='OwnerText' idField='Id' groupIDField='OwnerGroupId'
                            colorField='OwnerColor'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</div>
</template>
<script>
    import Vue from 'vue';
    import { resourceData } from './datasource.js';
    import { SchedulePlugin, Month, Week, Agenda, TimelineViews,
TimelineMonth } from '@syncfusion/ej2-vue-schedule';
    import { RadioButtonPlugin} from '@syncfusion/ej2-vue-buttons';
    Vue.use(SchedulePlugin);
    Vue.use(RadioButtonPlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 3, 4),
```

```
                group: {
                    resources: ['Rooms', 'Owners']
                },
                roomDataSource: [
                    { RoomText: 'ROOM 1', Id: 1, RoomGroupId: 1, RoomColor:
'#cb6bb2' },
                    { RoomText: 'ROOM 2', Id: 2, RoomGroupId: 2, RoomColor:
'#56ca85' }
                ],
                allowMultiple: true,
                ownerDataSource: [
                    { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1,
OwnerColor: '#ffaa00' },
                    { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2,
OwnerColor: '#f8a398' },
                    { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1,
OwnerColor: '#7499e1' }
                ],
                eventSettings: { dataSource: resourceData,
resourceColorField: 'Rooms' }
            }
        },
        provide: {
            schedule: [Month, Week, Agenda, TimelineViews, TimelineMonth]
        },
        methods: {
            onChange(args) {

this.$refs.scheduleObj.ej2Instances.eventSettings.resourceColorField =
args.value;
            }
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/resource-color-cs1" %}

The value of the resourceColorField field should be mapped with the name value given within the resources property.

Copyright © 2001 -2024 Syncfusion Inc.

Dynamically add and remove resources

It is possible to add or remove the resources dynamically to and from the Scheduler respectively. In the following example, when the checkboxes are checked and unchecked, the respective resources gets added up or removed from the Scheduler layout. To add new resource dynamically, addResource method is used which accepts the arguments such as resource object, resource name (within which level, the resource object to be added) and index (position where the resource needs to be added).

To remove the resources dynamically, removeResource method is used which accepts the index (position from where the resource to be removed) and resource name (within which level, the resource object presents) as parameters.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div class='col-lg-3'>
                <table id='property' title='Show / Hide Resource'
style='width: 100%;padding-right:25px'>
                    <tbody>
                        <tr style='height: 50px'>
                                <td style='width: 100%'>
                                    <ejs-checkbox cssClass='personal'
label='My Calender' value='1' :checked='true' :disabled='true'
:change='onChange'></ejs-checkbox>
                                </td>
                            </tr>
                            <tr style='height: 50px'>
                                <td style='width: 100%'>
                                    <ejs-checkbox cssClass='company'
label='Company' value='2' :checked='false' :change='onChange'></ejs-
checkbox>
                                </td>
                            </tr>
                            <tr style='height: 50px'>
                                <td style='width: 100%'>
                                    <ejs-checkbox cssClass='birthday'
label='Birthday' value='3' :checked='false' :change='onChange'></ejs-
checkbox>
                                </td>
                            </tr>
                            <tr style='height: 50px'>
                                <td style='width: 100%'>
                                    <ejs-checkbox cssClass='holiday'
label='Holiday' value='4' :checked='false' :change='onChange'></ejs-
checkbox>
                                </td>
                            </tr>
                        </tbody>
                </table>
            </div>
            <div id='container'>
                <ejs-schedule id='Schedule' ref='ScheduleObj' width='100%'
height='550px' :group='group' :selectedDate='selectedDate'
                    :eventSettings='eventSettings'>
                        <e-resources>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                        <e-resource field='CalendarId' title='Calendars'
:dataSource='resourceDataSource' :allowMultiple='allowMultiple'
                         name='Calendars' textField='CalendarText'
idField='CalendarId' colorField='CalendarColor'>
                        </e-resource>
                    </e-resources>
                    <e-views>
                        <e-view option='Month'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { holidayData, birthdayData, companyData, personalData } from
'./datasource.js';
    import { SchedulePlugin, Month, TimelineMonth } from '@syncfusion/ej2-
vue-schedule';
    import { CheckBoxPlugin } from '@syncfusion/ej2-vue-buttons';
    Vue.use(SchedulePlugin);
    Vue.use(CheckBoxPlugin);
    var calendarCollections = [
        { CalendarText: 'My Calendar', CalendarId: 1, CalendarColor:
'#c43081' },
        { CalendarText: 'Company', CalendarId: 2, CalendarColor: '#ff7f50'
},
        { CalendarText: 'Birthday', CalendarId: 3, CalendarColor: '#AF27CD'
},
        { CalendarText: 'Holiday', CalendarId: 4, CalendarColor: '#808000' }
    ];
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 3, 1),
                group: { resources: ['Calendars'] },
                resourceDataSource: [calendarCollections[0]],
                allowMultiple: true,
                eventSettings: { dataSource: this.generateCalendarData() }
            }
        },
        provide: {
            schedule: [Month, TimelineMonth]
        },
        methods: {
            generateCalendarData: function () {
                var collections = [];
                var dataCollections = [personalData, companyData,
birthdayData, holidayData];
                for (var i = 0; i < dataCollections.length; i++) {
                    collections = collections.concat(dataCollections[i]);
                }
                return collections;
            },
            onChange: function (args) {
```

```
                let scheduleObj = this.$refs.ScheduleObj;
                let value =
parseInt((args.event.currentTarget).querySelector('input').getAttribute('val
ue'), 10);
                let resourceData = calendarCollections.filter(function
(calendar) { return calendar.CalendarId === value; });
                if (args.checked) {
                    scheduleObj.addResource(resourceData[0], 'Calendars',
value - 1);
                    scheduleObj.dataBind();
                } else {
                    scheduleObj.removeResource(value, 'Calendars');
                    scheduleObj.dataBind();
                }
            }
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
  #app {
    display: flex;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/dynamic-resource-cs1" %}

Setting different working days and hours for resources

Each resource in the Scheduler can have different working hours as well as different working days set to it. There are default options available within the `resources` collection, to customize the default working hours and days of the Scheduler.

Set different work days

Different working days can be set for the resources of Scheduler using the `workDaysField` property which maps the working days field from the resource dataSource. This field accepts the collection of day indexes (from 0 to 6) of a week. By default, it is set to [1, 2, 3, 4, 5] and in the following example, each resource has been set with different values and therefore each of them will render only those working days. This option is applicable only on the calendar views and is not applicable on timeline views.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate'
:currentView='currentView' :group='group'>
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                        <e-view option='Month'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='ConferenceId' title='Conference'
name='Conferences' :allowMultiple='allowMultiple'
:dataSource='ownerDataSource'
                        textField='text' idField='id' colorField='color'
workDaysField='workDays'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { doctorData } from './datasource.js';
    import { SchedulePlugin, Week, WorkWeek, Month} from '@syncfusion/ej2-
vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 3, 4),
                currentView: 'WorkWeek',
                group: {
                    resources: ['Conferences']
                },
                allowMultiple: true,
                ownerDataSource: [
                    { text: 'Will Smith', id: 1, color: '#ea7a57', workDays:
[1, 2, 4, 5] },
                    { text: 'Alice', id: 2, color: 'rgb(53, 124, 210)',
workDays: [1, 3, 5] },
                    { text: 'Robson', id: 3, color: '#7fa900' , workDays:
[2,6]}
                ],
                eventSettings: { dataSource: doctorData },
            }
        },
        provide: {
            schedule: [Week, WorkWeek, Month]
        }
    }
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/multiple-resources-cs2" %}

Set different work hours

Working hours indicates the work hour duration of a day, which is highlighted visually with active color over the work cells. Each resource on the Scheduler can be defined with its own set of working hours as depicted in the following example.

- startHourField - Denotes the start time of the working/business hour in a day.
- endHourField - Denotes the end time limit of the working/business hour in a day.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate' :group='group'>
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='ConferenceId' title='Conference'
name='Conferences' :allowMultiple='allowMultiple'
:dataSource='ownerDataSource'
                        textField='text' idField='id' colorField='color'
startHourField='startHour'
                        endHourField='endHour'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
```

```
<script>
    import Vue from 'vue';
    import { doctorData } from './datasource.js';
    import { SchedulePlugin, Week, Month, TimelineViews, TimelineMonth} from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 3, 4),
                group: {
                    resources: ['Conferences']
                },
                allowMultiple: true,
                ownerDataSource: [
                    { text: 'Will Smith', id: 1, color: '#ea7a57',
startHour: '08:00', endHour: '15:00' },
                    { text: 'Alice', id: 2, color: '#357cd2', startHour:
'10:00', endHour: '18:00'},
                    { text: 'Robson', id: 3, color: '#7fa900', startHour:
'06:00', endHour: '13:00' }
                ],
                eventSettings: { dataSource: doctorData },
            }
        },
        provide: {
            schedule: [Week, Month, TimelineViews, TimelineMonth]
        }
    }
</script>
<style>
  @import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/multiple-resources-cs3" %}

In this example, a resource named Will Smith is depicted with working hours ranging from 8.00 AM to 3.00 PM and is visually illustrated with active colors, whereas the other two resources have different working hours set.

Copyright © 2001 -2024 Syncfusion Inc.

Hide non-working days when grouped by date

In Scheduler, you can set custom work days for each resource and group the Scheduler by date to display these work days. By default, the Scheduler will show all days when it is grouped by date, even if they are not included in the custom work days for the resources. However, you can use the hideNonWorkingDays property to only display the custom work days in the Scheduler.

To use the hideNonWorkingDays property, you need to include it in the configuration options for your Scheduler component. Set the value of hideNonWorkingDays to true to enable this feature.

Example: To display the Scheduler with resources grouped by date for custom working days,

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:selectedDate='selectedDate' :group='group'>
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='OwnerId' title='Owner'
name='Owners' :allowMultiple='allowMultiple'
                            :dataSource='ownerDataSource' textField='text'
idField='id' colorField='color'
                            workDaysField='workDays'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Week, Month, Agenda } from '@syncfusion/ej2-vue-
schedule';
Vue.use(SchedulePlugin);
export default {
    data() {
        return {
            width: '100%',
            height: '550px',
            selectedDate: new Date(2018, 3, 4),
            group: {
                byDate: true,
                resources: ['Owners'],
                hideNonWorkingDays: true,
            },
            allowMultiple: true,
            ownerDataSource: [
                { text: 'Alice', id: 1, color: '#1aaa55', workDays: [1, 2,
3, 4] },
```

```
                { text: 'Smith', id: 2, color: '#7fa900', workDays: [2, 3,
5] }
            ],
        }
    },
    provide: {
        schedule: [Week, Month, Agenda]
    }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/single-level-resource-cs3" %}

The hideNonWorkingDays property only applies when the Scheduler is grouped byDate.

Compact view in mobile

Although the Scheduler views are designed keeping in mind the responsiveness of the control in mobile devices, however when using Scheduler with multiple resources - it is difficult to view all the resources and its relevant events at once on the mobile. Therefore, we have introduced a new compact mode specially for displaying multiple resources of Scheduler on mobile devices. By default, this mode is enabled while using Scheduler with multiple resources on mobile devices. If in case, you need to disable this compact mode, set `false` to the `enableCompactView` option within the `group` property. Disabling this option will display the exact desktop mode of Scheduler view on mobile devices.

With this compact view enabled on mobile, you can view only single resource at a time and to switch to other resources, there is a treeview at the left listing out all other available resources - clicking on which will display that particular resource and its related appointments.

Copyright © 2001 -2024 Syncfusion Inc.

Clicking on the menu icon before the resource text will show the resources available in the Scheduler as following.

Copyright © 2001 -2024 Syncfusion Inc.

Adaptive UI in desktop

By default, the Scheduler layout adapts automatically in the desktop and mobile devices with appropriate UI changes. In case, if the user wants to display the Adaptive scheduler in desktop mode with adaptive enhancements, then the property `enableAdaptiveUI` can be set to true. Enabling this option will display the exact mobile mode of Scheduler view on desktop devices.

Some of the default changes made for compact Scheduler to render in desktop devices are as follows,

- View options displayed in the Navigation drawer.
- Plus icon is added to the header for new event creation.
- Today icon is added to the header instead of the Today button.
- With Multiple resources – only one resource has been shown to enhance the view experience of resource events details clearly. To switch to other resources, there is a TreeView on the left that lists all other available resources, clicking on which will display that particular resource and its related events.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
```

```
            <ejs-schedule id="schedule" ref="ScheduleObj" width='100%'
height='650px'
                    :selectedDate="selectedDate"
:eventSettings="eventSettings" currentView="Month"
:enableAdaptiveUI="enableAdaptiveUI" :group="group">
                    <e-views>
                        <e-view option="Day"></e-view>
                        <e-view option="Week"></e-view>
                        <e-view option="Month"></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='ProjectId' title='Choose Project'
name='Projects' :dataSource='projectResourceDataSource' textField='text'
                            idField='id' colorField='color'>
                        </e-resource>
                        <e-resource field='TaskId' title='Employee'
name='Employees' :dataSource='employeeDataSource'
:allowMultiple='allowMultiple'
                            textField='text' idField='id'
groupIDField='groupId' colorField='color'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { resourceData, timelineResourceData } from './datasource.js';
    import { SchedulePlugin, Day, Week, Month, DragAndDrop, Resize } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: {
                    dataSource: this.generateData()
                },
                selectedDate: new Date(2018, 3, 4),
                enableAdaptiveUI: true,
                allowMultiple : true,
                group: {
                    resources: ['Projects', 'Employees']
                },
                projectResourceDataSource: [
                    { text: 'PROJECT 1', id: 1, color: '#cb6bb2' },
                    { text: 'PROJECT 2', id: 2, color: '#56ca85' },
                    { text: 'PROJECT 3', id: 3, color: '#df5286' }
                ],
                employeeDataSource: [
                    { text: 'Nancy', id: 1, groupId: 1, color: '#df5286' },
                    { text: 'Steven', id: 2, groupId: 1, color: '#7fa900' },
                    { text: 'Robert', id: 3, groupId: 2, color: '#ea7a57' },
                    { text: 'Smith', id: 4, groupId: 2, color: '#5978ee' },
                    { text: 'Micheal', id: 5, groupId: 3, color: '#df5286'
},
                    { text: 'Root', id: 6, groupId: 3, color: '#00bdae' }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                ],
            }
        },
        provide: {
            schedule: [Day, Week, Month, DragAndDrop, Resize]
        },
        methods: {
            generateData: function () {
                var collections = [];
                var dataCollections = [resourceData, timelineResourceData];
                for (var i = 0; i < dataCollections.length; i++) {
                    collections = collections.concat(dataCollections[i]);
                }
                return collections;
            }
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/resource-grouping-cs7" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Header rows in Vue Schedule component

The Timeline views can have additional header rows other than its default date and time header rows. It is possible to show individual header rows for displaying year, month and week separately using the headerRows property. This property is applicable only on the timeline views. The possible rows which can be added using headerRows property are as follows.

- Year
- Month
- Week
- Date
- Hour

Copyright © 2001 -2024 Syncfusion Inc.

The Link to the Video row is not applicable for Timeline month view.

Check out this video to learn about customizing header rows of the timeline views in Vue Scheduler.

The following example shows the Scheduler displaying all the available header rows on timeline views.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :eventSettings='eventSettings' :views='views'
:startHour='startHour' :endHour='endHour' :headerRows='headerRows'></ejs-
schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, TimelineViews } from '@syncfusion/ej2-vue-
schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      startHour: '09:00',
      endHour: '13:00',
      headerRows: [
        { option: 'Year' },
        { option: 'Month' },
        { option: 'Week' },
        { option: 'Date' },
        { option: 'Hour' }
      ],
      views: ['TimelineWeek'],
      selectedDate: new Date(2018, 11, 31),
      eventSettings: { dataSource: scheduleData }
    }
  },
  provide: {
    schedule: [TimelineViews]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/header-rows-cs1" %}

Display year and month rows in timeline views

To display the timeline Scheduler simply with year and month names alone, define the option Year and Month within the headerRows property.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :eventSettings='eventSettings' :views='views'
:currentView='currentView' :headerRows='headerRows'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, TimelineMonth } from '@syncfusion/ej2-vue-
schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      width: '100%',
      height: '550px',
      selectedDate: new Date(2018, 11, 31),
      headerRows: [{ option: 'Year' }, { option: 'Month' }],
      currentView: 'TimelineMonth',
      views: [{ option: 'TimelineMonth', interval: 24 }],
      eventSettings: { dataSource: scheduleData }
    }
  },
  provide: {
    schedule: [TimelineMonth]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/header-rows-cs2" %}

Display week numbers in timeline views

The week number can be displayed in a separate header row of the timeline Scheduler by setting Week option within headerRows property.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :eventSettings='eventSettings' :views='views'
:currentView='currentView' :headerRows='headerRows'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, TimelineMonth, TimelineViews } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      width: '100%', height: '550px',
      selectedDate: new Date(2018, 11, 31),
      headerRows: [{ option: 'Week' }, { option: 'Date' }, { option: 'Hour'
}],
      currentView: 'TimelineMonth',
      views: [
          { option: 'TimelineMonth', interval: 24 },
          { option: 'TimelineWeek', interval: 3 },
          { option: 'TimelineDay', interval: 4 }
      ],
      eventSettings: { dataSource: scheduleData }
    }
  },
  provide: {
    schedule: [TimelineViews, TimelineMonth]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
```

```
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/header-rows-cs3" %}

Timeline view displaying dates of a complete year

It is possible to display a complete year in a timeline view by setting interval value as 12 and defining **TimelineMonth** view option within the views property of Scheduler.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :eventSettings='eventSettings' :views='views'
:headerRows='headerRows'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, TimelineMonth, TimelineViews } from
'@syncfusion/ej2-vue-schedule';
import { eventData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      width: '100%', height: '550px',
      selectedDate: new Date(2018, 0, 1),
      headerRows: [{ option: 'Month' }, { option: 'Date' }],
      views: [{ option: 'TimelineMonth', interval: 12 }],
      eventSettings: { dataSource: eventData }
    }
  },
  provide: {
    schedule: [TimelineMonth]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/header-rows-cs4" %}

Customizing the header rows using template

You can customize the text of the header rows and display any images or formatted text on each individual header rows using the built-in template option available within the headerRows property.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :eventSettings='eventSettings' :views='views'
:headerRows='headerRows'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, TimelineMonth, getWeekNumber } from
'@syncfusion/ej2-vue-schedule';
import { Internationalization } from '@syncfusion/ej2-base';
import { eventData } from './datasource.js';
Vue.use(SchedulePlugin);
var instance = new Internationalization();
var yearHeaderVue = Vue.component('year-header', {
    template: '<span class="year">{{getYearDetails(data)}}</span>',
    data() {
        return {
            data: {}
        };
    },
    methods: {
        getYearDetails: function (value) {
            return 'Year: ' + instance.formatDate(value.date, { skeleton:
'y' });
        }
    }
});
var monthHeaderVue = Vue.component('month-header', {
```

```
    template: '<span class="month">{{getMonthDetails(data)}}</span>',
    data() {
        return {
            data: {}
        };
    },
    methods: {
        getMonthDetails: function (value) {
            return 'Month ' + instance.formatDate(value.date, { skeleton:
'M' });
        }
    }
});
var weekHeaderVue = Vue.component('week-header', {
    template: '<span class="week">{{getWeekDetails(data)}}</span>',
    data() {
        return {
            data: {}
        };
    },
    methods: {
        getWeekDetails: function (value) {
            return 'Week: ' + getWeekNumber(value.date);
        }
    }
});
export default {
  data (){
    return {
      width: '100%',
      height: '550px',
      selectedDate: new Date(2018, 0, 1),
      headerRows: [
        {
            option: 'Year',
            template: function (e) {
                return {
                    template: yearHeaderVue
                };
            }
        },
        {
            option: 'Month',
            template: function (e) {
                return {
                    template: monthHeaderVue
                };
            },
        },
        {
            option: 'Week',
            template: function (e) {
                return {
                    template: weekHeaderVue
                };
            }
        },
```

```
        { option: 'Date' }
      ],
      views: [{ option: 'TimelineMonth' }],
      eventSettings: { dataSource: eventData }
    }
  },
  provide: {
    schedule: [TimelineMonth]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/header-rows-cs5" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Row auto height in Vue Schedule component

By default, the height of the Scheduler rows in Timeline views are static and therefore, when the same time range holds multiple overlapping appointments, a `+n more` text indicator will be displayed. With this feature enabled, you can now view all the overlapping appointments present in those specific time range by auto-adjusting the row height based on the presence of the appointments count, instead of displaying the `+n more` text indicators.

To enable auto row height adjustments on Scheduler Timeline views and Month view, set `true` to the rowAutoHeight property whose default value is `false`.

This auto row height adjustment is applicable only on all the Timeline views as well as on the calendar Month view.

Now, let's see how it works on those applicable views with examples.

Calendar month view

By default, the rows of the calendar Month view can hold only the limited appointments count based on its row height, and the rest of the overlapping appointments are indicated with a `+n more` text indicator. The following example shows how the month view row auto-adjusts based on the number of appointments count, when this `RowAutoHeight` feature is enabled.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:rowAutoHeight='rowAutoHeight'>
                    <e-views>
                        <e-view option='Month'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Month } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                currentView: 'Month',
                rowAutoHeight: true,
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Month]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/working-days-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Timeline views

When the feature RowAutoHeight is enabled in Timeline views, the row height gets auto-adjusted based on the number of overlapping events occupied on the same time range, which is demonstrated in the following example.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:rowAutoHeight='rowAutoHeight'>
                    <e-views>
                        <e-view option='TimelineDay'></e-view>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineWorkWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, TimelineViews, TimelineMonth, Agenda
    } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                currentView: 'TimelineWeek',
                rowAutoHeight: true,
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [TimelineViews, TimelineMonth, Agenda]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/working-days-cs2" %}

Timeline views with multiple resources

The following example shows how the auto row adjustment feature works on timeline views with multiple resources.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:rowAutoHeight='rowAutoHeight' :group='group'>
                    <e-views>
                        <e-view option='TimelineDay'></e-view>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineWorkWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field='RoomId' title='Room Type'
name='MeetingRoom' allowMultiple=true :dataSource='meetingRoomDataSource'
textField='text' idField='id' colorField='color'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { roomData } from './datasource.js';
    import { SchedulePlugin, TimelineViews, TimelineMonth, Agenda
    } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 7, 1),
                currentView: 'TimelineWeek',
                rowAutoHeight: true,
                eventSettings: {
                    dataSource: roomData,
                    fields: {
```

```
                        id: 'Id',
                        subject: { name: 'Subject', title: 'Summary' },
                        location: { name: 'Location', title: 'Location' },
                        description: { name: 'Description', title:
'Comments' },
                        startTime: { name: 'StartTime', title: 'From' },
                        endTime: { name: 'EndTime', title: 'To' }
                    }
                },
                group: {
                    enableCompactView: false,
                    resources: ['MeetingRoom']
                },
                meetingRoomDataSource: [
                    { text: 'Room A', id: 1, color: '#98AFC7' },
                    { text: 'Room B', id: 2, color: '#99c68e' },
                    { text: 'Room C', id: 3, color: '#C2B280' },
                    { text: 'Room D', id: 4, color: '#3090C7' },
                    { text: 'Room E', id: 5, color: '#95b9' },
                    { text: 'Room F', id: 6, color: '#95b9c7' },
                    { text: 'Room G', id: 7, color: '#deb887' },
                    { text: 'Room H', id: 8, color: '#3090C7' },
                    { text: 'Room I', id: 9, color: '#98AFC7' },
                    { text: 'Room J', id: 10, color: '#778899' }
                ]
            }
        },
        provide: {
            schedule: [TimelineViews, TimelineMonth, Agenda]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/working-days-cs3" %}

Appointments occupying entire cell

By default, with the feature rowAutoHeight, there will be a space in the bottom of the cell when appointment is rendered. To avoid this space, we can set true to the property ignoreWhitespace with in

eventSettings whereas its default property value is false. In the following code example, the whitespace below the appointments has been ignored.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate'
:currentView='currentView' :group='group' :rowAutoHeight='rowAutoHeight'>
                    <e-views>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                    </e-views>
                    <e-resources>
                        <e-resource field= 'RoomId' title= 'Room' name=
'Rooms' :dataSource='roomDataSource' textField='RoomText' idField='Id'
colorField='RoomColor'>
                        </e-resource>
                        <e-resource field='OwnerId' title='Owner'
name='Owners' :allowMultiple='allowMultiple' :dataSource='ownerDataSource'
                        textField='OwnerText' idField='Id' groupIDField=
'OwnerGroupId' colorField='OwnerColor'>
                        </e-resource>
                    </e-resources>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { resourceData } from './datasource.js';
    import { SchedulePlugin, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                width: '100%',
                height: '550px',
                currentView: 'TimelineWeek',
                rowAutoHeight: true,
                selectedDate: new Date(2021, 7, 4),
                group: {
                    resources: ['Rooms', 'Owners']
                },
                roomDataSource: [
                    { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2' },
                    { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
                ],
                allowMultiple: true,
                ownerDataSource: [
                    { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1,
OwnerColor: '#ffaa00' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2,
OwnerColor: '#f8a398' },
                    { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1,
OwnerColor: '#7499e1' }
                ],
                eventSettings: { dataSource: resourceData, ignoreWhitespace:
true },
            }
        },
        provide: {
            schedule: [TimelineViews, TimelineMonth]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/ignore-whitespace-cs1" %}

Note: The property `ignoreWhitespace` will be applicable only when `rowAutoHeight` feature is enabled in the Scheduler

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Header bar in Vue Schedule component

The header part of Scheduler can be customized easily with the built-in options available.

Show or Hide header bar

By default, the header bar holds the date and view navigation options, through which the user can switch between the dates and various views. This header bar can be hidden from the UI by setting `false` to the showHeaderBar property. It's default value is `true`.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedData'
:eventSettings='eventSettings' :showHeaderBar='showHeaderBar'></ejs-
schedule>
```

```
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek } from '@syncfusion/ej2-vue-
schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      showHeaderBar: false,
      eventSettings: { dataSource: scheduleData  },
      selectedData: new Date(2018, 1, 15),
      views: ['Day', 'Week', 'WorkWeek'],
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/header-bar-cs1" %}

Customizing header bar using template

Apart from the default date navigation and view options available on the header bar, you can add custom items into the Scheduler header bar by making use of the toolbarItems property. To display the default items, it's essential to assign a name field to each item. The names of the default items are `Previous`, `Next`, `Today`, `DateRangeText`, `NewEvent`, and `Views`. For custom items you can give the name as `Custom` to the `name` field. Here, the default items such as previous, next, date range text, and today have been used along with external icon as custom items.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <div id='app'>
    <div id='container'>
      <ejs-schedule id='Schedule' :height='height'
:selectedDate='selectedData' :eventSettings='eventSettings'
        :views='views'>
          <e-resources>
            <e-resource field="OwnerId" title="Owners"
:dataSource="resourceDataSource" :query='resourceQuery'
:allowMultiple="allowMultiple" name="Owners"
              textField="OwnerText" idField="OwnerId" colorField='Color'>
            </e-resource>
          </e-resources>
        <e-toolbaritems>
          <e-toolbaritem name='Previous' align='Left'></e-toolbaritem>
          <e-toolbaritem name='Next' align='Left'></e-toolbaritem>
          <e-toolbaritem name='DateRangeText' align='Left'></e-toolbaritem>
          <e-toolbaritem :template="'Template'">
            <template v-slot:Template>
              <ejs-dropdownlist id='dropdownlist' :value="dropdownvalue"
:change="onResourceChange" width="125px" :showClearButton=false
:dataSource='resourceDataSource' :fields='fields'></ejs-dropdownlist>
            </template>
          </e-toolbaritem>
          <e-toolbaritem name='Today' align='Right'></e-toolbaritem>
        </e-toolbaritems>
      </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ScheduleComponent, Month, ToolbarItemsDirective,
ToolbarItemDirective, ResourceDirective, ResourcesDirective } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
import { DropDownListComponent } from "@syncfusion/ej2-vue-dropdowns";
import {Predicate, Query } from '@syncfusion/ej2-data';
export default {
  components: {
    'ejs-schedule': ScheduleComponent,
    'e-resource': ResourceDirective,
    'e-resources': ResourcesDirective,
    'e-toolbaritems': ToolbarItemsDirective,
    'e-toolbaritem': ToolbarItemDirective,
    'ejs-dropdownlist': DropDownListComponent
  },
  data() {
    return {
      height: '550px',
      width: '100%',
      eventSettings: { dataSource: scheduleData, query: new
Query().where('OwnerId', 'equal', 1) },
      selectedData: new Date(2023, 10, 15),
      views: ['Month'],
      currentView: 'Month',
      resourceDataSource: [
```

```
        { OwnerText: 'Margaret', OwnerId: 1, Color: '#ea7a57' },
        { OwnerText: 'Robert', OwnerId: 2, Color: '#df5286' },
        { OwnerText: 'Laura', OwnerId: 3, Color: '#865fcf' }
      ],
      fields: { text: 'OwnerText', value: 'OwnerId' },
      dropdownvalue: 1,
      resourceQuery: new Query().where('OwnerId', 'equal', 1),
      allowMultiple: true,
    }
  },
  provide: {
    schedule: [Month]
  },
  methods: {
    onResourceChange(args) {
      let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
      let resourcePredicate;
      let value = args.value;
      resourcePredicate = new Predicate('OwnerId', 'equal', value)
      scheduleObj.resources[0].query = resourcePredicate ? new
Query().where(resourcePredicate) :
        new Query().where('OwnerId', 'equal', 1);
      scheduleObj.eventSettings.query = new Query().where('OwnerId',
'equal', value);
    },
  },
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/header-bar-template" %}

Customizing header bar using events

Apart from the default date navigation and view options available on the header bar, you can add custom items into the Scheduler header bar by making use of the actionBegin event. Here, an employee image is added to the header bar, clicking on which will open the popup showing that person's short profile information.

APP.VUE

```
<template>
```

```
  <div id='app'>
    <div id='container'>
        <ejs-schedule id='Schedule' :height='height'
:selectedDate='selectedData' :eventSettings='eventSettings' :views='views'
:actionBegin='onActionBegin' :actionComplete='onActionComplete'></ejs-
schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { createElement, compile } from '@syncfusion/ej2-base';
import { Popup } from '@syncfusion/ej2-popups';
import { ItemModel } from '@syncfusion/ej2-navigations';
import { SchedulePlugin, Month } from '@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
var headerTemplateVue = Vue.component('header-template', {
    template: `<div class="profile-container"><div class="profile-
image"></div><div class="content-wrap">
        '<div class="name">Nancy</div><div class="destination">Product
Manager</div><div class="status"><div class="status-
icon"></div>Online</div></div></div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      eventSettings: { dataSource: scheduleData  },
      selectedData: new Date(2018, 1, 15),
      views: ['Month'],
      currentView: 'Month',
      headerTemplate: function () {
        return { template: headerTemplateVue }
      },
    }
  },
  methods: {
      onActionBegin: function(args) {
        if (args.requestType === 'toolbarItemRendering') {
            let userIconItem = {
                align: 'Right', prefixIcon: 'user-icon', text: 'Nancy',
cssClass: 'e-schedule-user-icon'
            };
            args.items.push(userIconItem);
        }
      },
      onActionComplete: function(args) {
        let scheduleElement = document.getElementById('Schedule');
        if (args.requestType === 'toolBarItemRendered') {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            let userIconEle = scheduleElement.querySelector('.e-schedule-
user-icon');
            userIconEle.onclick = () => {
                this.profilePopup.relateTo = userIconEle;
                this.profilePopup.dataBind();
                if (this.profilePopup.element.classList.contains('e-popup-
close')) {
                    this.profilePopup.show();
                } else {
                    this.profilePopup.hide();
                }
            };
        }
        let userContentEle = createElement('div', {
            className: 'e-profile-wrapper'
        });
        scheduleElement.parentElement.appendChild(userContentEle);
        let userIconEle = scheduleElement.querySelector('.e-schedule-user-
icon');
        let getDOMString = compile(this.headerTemplate);
        let output = getDOMString({});
        this.profilePopup = new Popup(userContentEle, {
            content: output[0],
            relateTo: userIconEle,
            position: { X: 'left', Y: 'bottom' },
            collision: { X: 'flip', Y: 'flip' },
            targetType: 'relative',
            viewPortElement: scheduleElement,
            width: 210,
            height: 80
        });
        this.profilePopup.hide();
      }
  },
  provide: {
    schedule: [Month]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
.e-schedule .e-schedule-toolbar .user-icon,
.e-profile-wrapper .profile-image {
```

```
  background-image:
url('https://ej2.syncfusion.com/demos/src/schedule/images/nancy.png');
  background-position: center center;
  background-repeat: no-repeat;
  background-size: cover;
  border-radius: 50%;
}
/* csslint ignore:start */
.e-schedule .e-schedule-toolbar .user-icon {
  height: 24px;
  min-width: 24px !important;
  width: 24px !important;
}
/* csslint ignore:end */
.e-schedule .e-schedule-toolbar .e-toolbar-items .e-schedule-user-icon .e-
tbar-btn:hover {
  background-color: inherit;
}
.e-schedule .e-schedule-toolbar .e-toolbar-items .e-schedule-user-icon .e-
tbar-btn-text {
  display: none;
}
/* csslint ignore:start */
.e-schedule .e-schedule-toolbar .e-toolbar-pop .e-schedule-user-icon .e-
tbar-btn-text {
  padding-left: 8px !important;
}
/* csslint ignore:end */
.e-profile-wrapper {
  width: 210px;
  height: 80px;
  background-color: #fafafa;
  box-shadow: inset 0 0 5px rgba(0, 0, 0, 0.2);
  overflow: hidden;
}
.e-profile-wrapper .profile-container {
  display: flex;
  padding: 10px;
}
.e-profile-wrapper .profile-image {
  box-shadow: inset 0 0 1px #e0e0e0, inset 0 0 14px rgba(0, 0, 0, 0.2);
  width: 60px;
  height: 60px;
}
.e-profile-wrapper .content-wrap {
  padding-left: 10px;
}
.e-profile-wrapper .name {
  font-size: 14px;
  line-height: 20px;
  font-weight: 500;
  margin-top: 2px;
}
.e-profile-wrapper .destination {
  font-size: 12px;
}
.e-profile-wrapper .status-icon {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  height: 6px;
  width: 6px;
  background: green;
  border-radius: 100%;
  float: left;
  margin-right: 4px;
  margin-top: 4px;
 }
 .e-profile-wrapper .status {
  font-size: 11px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/header-bar-cs2" %}

How to display the view options within the header bar popup

By default, the header bar holds the view navigation options, through which the user can switch between various views. You can move this view options to the header bar popup by setting true to the enableAdaptiveUI property.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedData'
:eventSettings='eventSettings' :enableAdaptiveUI='enableAdaptiveUI'></ejs-
schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      enableAdaptiveUI: true,
      eventSettings: { dataSource: scheduleData  },
      selectedData: new Date(2018, 1, 15),
      views: ['Day', 'Week', 'WorkWeek', 'Month', 'Agenda,'],
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/header-bar-cs3" %}

Refer here to know more about adaptive UI in resources scheduler.

Date header customization

The Scheduler UI that displays the date text on all views are considered as the date header cells. You can customize the date header cells of Scheduler either using dateHeaderTemplate or renderCell event.

Using date header template

The dateHeaderTemplate option is used to customize the date header cells of day, week and work-week views.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:dateHeaderTemplate='dateHeaderTemplate' :cssClass='cssClass'>
            <e-views>
                <e-view option='Day'></e-view>
                <e-view option='Week'></e-view>
                <e-view option='Agenda'></e-view>
                <e-view option='TimelineWorkWeek'></e-view>
                <e-view option='TimelineMonth'></e-view>
            </e-views>
        </ejs-schedule>
    </div>
  </div>
</template>
<script>
    import Vue from 'vue';
    import { Internationalization } from '@syncfusion/ej2-base';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, Agenda, TimelineViews, TimelineMonth
} from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    var instance = new Internationalization();
    var dateHeaderTemplateVue = Vue.component('demo', {
```

```
        template: '<div class="date-text" v-
html="getDateHeaderText(data.date)"></div>'+
        '<div v-html="getWeather(data.date)"></div>',
        data() {
            return {
                data: {}
            };
        },
        methods: {
            getDateHeaderText: function (Date) {
                return instance.formatDate(Date, { skeleton: 'Ed' });
            },
            getWeather: function (Date) {
                switch (Date.getDay()) {
                    case 0:
                        return '<div class="weather-text">25°C</div>';
                    case 1:
                        return '<div class="weather-text">18°C</div>';
                    case 2:
                        return '<div class="weather-text">10°C</div>';
                    case 3:
                        return '<div class="weather-text">16°C</div>';
                    case 4:
                        return '<div class="weather-text">8°C</div>';
                    case 5:
                        return '<div class="weather-text">27°C</div>';
                    case 6:
                        return '<div class="weather-text">17°C</div>';
                    default:
                        return null;
                }
            }
        }
    });
    export default {
        data () {
            return {
                eventSettings: { dataSource: scheduleData },
                cssClass: 'schedule-date-header-template',
                selectedDate: new Date(2018, 1, 15),
                dateHeaderTemplate: function (e) {
                    return { template: dateHeaderTemplateVue }
                }
            }
        },
        provide: {
            schedule: [Day, Week, Agenda, TimelineViews, TimelineMonth]
        }
    }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/date-header-cs1" %}

Using renderCell event

In month view, the date header template is not applicable and therefore the same customization can be added beside the date text in month cells by making use of the renderCell event.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule id='Schedule' :height='height'
:selectedDate='selectedData' :eventSettings='eventSettings' :views='views'
:renderCell='onRenderCell'>
        </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { createElement } from '@syncfusion/ej2-base';
import { SchedulePlugin, Month } from '@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      eventSettings: { dataSource: scheduleData  },
      selectedData: new Date(2018, 1, 15),
      views: ['Month'],
      currentView: 'Month'
    }
  },
  methods: {
      onRenderCell: function(args) {
        if (args.elementType === 'monthCells') {
            let ele = document.createElement('div');
            ele.innerHTML = this.getWeather(args.date);
            (args.element).appendChild(ele.firstChild);
        }
      },
      getWeather: function(value) {
        switch (value.getDay()) {
            case 0:
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                return '<div class="weather-text">25°C</div>';
            case 1:
                return '<div class="weather-text">18°C</div>';
            case 2:
                return '<div class="weather-text">10°C</div>';
            case 3:
                return '<div class="weather-text">16°C</div>';
            case 4:
                return '<div class="weather-text">8°C</div>';
            case 5:
                return '<div class="weather-text">27°C</div>';
            case 6:
                return '<div class="weather-text">17°C</div>';
            default:
                return null;
        }
      }
  },
  provide: {
    schedule: [Month]
  }
}
</script>
<style>
  @import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
  .weather-text {
    float: right;
    margin: -20px 12px 0 0;
    width: 20px;
    height: 20px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/render-cell-cs1" %}

Customizing the date range text

The dateRangeTemplate option allows you to customize the text content of the date range displayed in the scheduler. By default, the date range text is determined by the scheduler view being used. However,

you can use the dateRangeTemplate option to override the default text and specify your own custom text to be displayed.

The dateRangeTemplate property includes startDate, endDate and currentView options, you can customize the date range text using these available options.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule height='550px' width='100%'
:selectedDate='selectedDate'
                :dateRangeTemplate='dateRangeTemplate'>
                <e-views>
                    <e-view option='Day'></e-view>
                    <e-view option='Week'></e-view>
                    <e-view option='Agenda'></e-view>
                    <e-view option='TimelineWorkWeek'></e-view>
                    <e-view option='TimelineMonth'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { Internationalization } from '@syncfusion/ej2-base';
import { SchedulePlugin, Day, Week, Agenda, TimelineViews, TimelineMonth }
from '@syncfusion/ej2-vue-schedule';
Vue.use(SchedulePlugin);
var instance = new Internationalization();
var dateRangeTemplateVue = Vue.component('demo', {
    template: '<div class="date-text">{{getDateRange(data.startDate)}}-
{{getDateRange(data.endDate)}}</div>',
    data() {
        return {
            data: {}
        };
    },
    methods: {
        getDateRange: function (value) {
            return instance.formatDate(value, { skeleton: 'Ed' });
        }
    }
});
export default {
    data() {
        return {
            selectedDate: new Date(2018, 1, 15),
            dateRangeTemplate: function (e) {
                return { template: dateRangeTemplateVue }
            }
        }
    },
    provide: {
        schedule: [Day, Week, Agenda, TimelineViews, TimelineMonth]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/date-header-cs2" %}

Customizing header indent cells

It is possible to customize the header indent cells using the `headerIndentTemplate` option and change the look and appearance in both the vertical and timeline views. In vertical views, You can customize the header indent cells at the hierarchy level and you can customize the resource header left indent cell in timeline views using the template option.

Example: To customize the header left indent cell to display resources text, refer to the below code example.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
      <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate' :group='group'>
        <e-views>
          <e-view option='Day'></e-view>
          <e-view option='Week'></e-view>
          <e-view option='TimelineWeek'></e-view>
          <e-view option='TimelineMonth'></e-view>
        </e-views>
        <e-resources>
          <e-resource field='OwnerId' title='Owner' name='Owners'
:allowMultiple='allowMultiple' :dataSource='ownerDataSource'
            textField='OwnerText' idField='Id' colorField='OwnerColor'>
          </e-resource>
        </e-resources>
      </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { SchedulePlugin, Week, Day, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-vue-schedule';
import { resourceData } from './datasource.js';
Vue.use(SchedulePlugin);
var headerIndentTemplate = Vue.component('headerTemplate', {
        template: `<div class='e-resource-text'>
          <div class="text">Resources</div>`,
        data() {
            return {
                data: {}
            };
        }
});
export default {
        data () {
            return {
                width: '100%',
                height: '550px',
                currentView: 'Week',
                selectedDate: new Date(2018, 3, 1),
                group: {
                    resources: ['Owners']
                },
                allowMultiple: true,
                headerIndentTemplate: function(e){
                    return {
                        template: headerIndentTemplate
                    };
                },
                ownerDataSource: [
                    { OwnerText: 'Nancy', Id: 1, OwnerColor: '#ffaa00' },
                    { OwnerText: 'Steven', Id: 2, OwnerColor: '#f8a398' },
                    { OwnerText: 'Michael', Id: 3, OwnerColor: '#7499e1' }
                ],
                eventSettings: { dataSource: resourceData },
            }
        },
        provide: {
            schedule: [Day, Week, TimelineViews, TimelineMonth]
        }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
.e-schedule .e-timeline-view .e-resource-left-td {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  vertical-align: bottom;
}
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-text,
.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-text {
  font-weight: 500;
  padding: 0;
}
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-text > div {
  border-right: 1px solid rgba(0, 0, 0, 0.12);
  border-top: 1px solid rgba(0, 0, 0, 0.12);
  flex: 0 0 33.3%;
  font-weight: 500;
  height: 36px;
  line-height: 34px;
  padding-left: 50px;
}
.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-text >
div {
  border-right: 1px solid rgba(0, 0, 0, 0.12);
  flex: 0 0 33.3%;
  font-weight: 500;
  height: 36px;
  line-height: 34px;
  padding-left: 50px;
}
/* csslint ignore:start */
.e-schedule .e-vertical-view .e-left-indent-wrap table tbody td.e-resource-
cells {
  border-bottom-color: rgba(0, 0, 0, 0.12);
}
.e-schedule .e-vertical-view .e-left-indent-wrap table tbody td.e-resource-
cells .e-resource-text {
  font-weight: 500;
}
.e-schedule .e-vertical-view .e-left-indent-wrap table tbody td.e-header-
cells .e-resource-text,
.e-schedule .e-vertical-view .e-left-indent-wrap table tbody td.e-all-day-
cells .e-resource-text {
  display: none;
}
/* csslint ignore:end */
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/header-indent-cs1" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Timescale in Vue Schedule component

The time slots are usually the time cells that are displayed on the Day, Week and Work Week views of both the calendar (to the left most position) and timeline views (at the top position). The timeScale property allows you to control and set the required time slot duration for the work cells displayed on Scheduler. It includes the following sub-options such as,

- enable - When set to true, allows the Scheduler to display the appointments accurately against the exact time duration. If set to false, all the appointments of a day will be displayed one below the other with no grid lines displayed. Its default value is true.
- interval – Defines the time duration on which the time axis to be displayed either in 1 hour or 30 minutes interval and so on. It accepts the values in minutes and defaults to 60.
- slotCount – Decides the number of slot count to be split for the specified time interval duration. It defaults to 2, thus displaying two slots to represent an hour(each slot depicting 30 minutes duration).

Note: The upper limit for rendering slots within a single day, utilizing the **interval** and **slotCount** properties of the **timeScale**, stands at 1000. This constraint aligns with the maximum **colspan** value permissible for the **table** element, also capped at 1000. This particular restriction is relevant exclusively to the TimelineDay, TimelineWeek and TimelineWorkWeek views.

Setting different time slot duration

The interval and slotCount properties can be used together on the Scheduler to set different time slot duration which is depicted in the following code example. Here, six time slots together represents an hour.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedData'
:eventSettings='eventSettings'
        :timeScale='timeScale'>
        <e-views>
          <e-view option='Day'></e-view>
          <e-view option='Week'></e-view>
          <e-view option='WorkWeek'></e-view>
        </e-views>
        </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, View, TimeScaleModel } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
import { extend } from '@syncfusion/ej2-base';
Vue.use(SchedulePlugin);
export default {
    data (){
      return {
        height: '550px',
        eventSettings: { dataSource: extend([], scheduleData, null, true)
},
        selectedData: new Date(2018, 1, 15),
        timeScale: {
          enable: true,
          interval: 60,
          slotCount: 6
```

```
        },
      }
    },
    provide: {
      schedule: [Day, Week, WorkWeek]
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/timescale-cs1" %}

Customizing time cells using template

The timeScale property also provides template option to allow customization of time slots which are as follows,

- majorSlotTemplate - The template option to be applied for major time slots. Here, the template accepts either the string or HTMLElement as template design and then the parsed design is displayed onto the time cells. The time details can be accessed within this template.
- minorSlotTemplate - The template option to be applied for minor time slots. Here, the template accepts either the string or HTMLElement as template design and then the parsed design is displayed onto the time cells. The time details can be accessed within this template.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule id='Schedule' :height='height'
:selectedDate='selectedData' :eventSettings='eventSettings'
        :timeScale='timeScale'>
        <e-views>
          <e-view option='Day'></e-view>
          <e-view option='Week'></e-view>
          <e-view option='WorkWeek'></e-view>
        </e-views>
        </ejs-schedule>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, View, TimeScaleModel } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
import { Internationalization, extend } from '@syncfusion/ej2-base';
Vue.use(SchedulePlugin);
  let instance = new Internationalization();
  var majorTemplateVue = Vue.component('demo', {
        template: '<div>{{majorSlotTemplate(data.date)}}</div>',
        data() {
            return {
                data: {}
            };
        },
        methods: {
            majorSlotTemplate: function (date) {
                return instance.formatDate(date, { skeleton: 'hm' });
            }
        }
    });
   var minorTemplateVue = Vue.component('demo', {
        template: '<div style="text-align: right; margin-right:
15px">{{minorSlotTemplate(data.date)}}</div>',
        data() {
            return {
                data: {},
                minorSlotTemplate: function (date) {
                return instance.formatDate(date, { skeleton: 'ms'
}).replace(':00', '');
                }
        };
      }
    });
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: extend([], scheduleData, null, true)  },
      selectedData: new Date(2018, 1, 15),
      majorSlotTemplate: function (e) {
          return { template: majorTemplateVue }
      },
      minorSlotTemplate: function (e) {
         return { template: minorTemplateVue }
      },
      timeScale: {
        enable: true,
        interval: 60,
        slotCount: 6
      },
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
  mounted: function () {
    let scheduleObj = document.getElementById('Schedule').ej2_instances[0];
    scheduleObj.timeScale.majorSlotTemplate = this.majorSlotTemplate;
    scheduleObj.timeScale.minorSlotTemplate = this.minorSlotTemplate;
    scheduleObj.dataBind();
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/timescale-cs2" %}

Hide the timescale

The grid lines which indicates the exact time duration can be enabled or disabled on the Scheduler, by setting `true` or `false` to the `enable` option within the timeScale property. It's default value is `true`.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedData'
:eventSettings='eventSettings'
        :timeScale='timeScale'>
        <e-views>
          <e-view option='Day'></e-view>
          <e-view option='Week'></e-view>
          <e-view option='TimelineWorkWeek'></e-view>
        </e-views>
        </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, TimelineViews, View } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
import { extend } from '@syncfusion/ej2-base';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: extend([], scheduleData, null, true)  },
      selectedData: new Date(2018, 1, 15),
      timeScale: {
        enable: false,
      },
    }
  },
  provide: {
    schedule: [Day, Week, TimelineViews]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/timescale-cs3" %}

Highlighting current date and time

By default, Scheduler indicates current date with a highlighted date header on all views, as well as marks accurately the system's current time on specific views such as Day, Week, Work Week, Timeline Day, Timeline Week and Timeline Work Week views. To stop highlighting the current time indicator on Scheduler views, set `false` to the showTimeIndicator property which defaults to `true`.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :selectedDate='selectedData'
:eventSettings='eventSettings' :showTimeIndicator = 'showTimeIndicator'>
          <e-views>
            <e-view option='Day'></e-view>
            <e-view option='Week'></e-view>
            <e-view option='WorkWeek'></e-view>
            <e-view option='TimelineDay'></e-view>
          </e-views>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, TimelineViews, View} from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
import { extend } from '@syncfusion/ej2-base';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      eventSettings: { dataSource: extend([], scheduleData, null, true)  },
      selectedData: new Date(2018, 1, 15),
      showTimeIndicator: false,
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, TimelineViews]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/timescale-cs4" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Working days in Vue Schedule component

The Scheduler can be customized on various aspects as well as it inherits almost all the calendar-specific features such as options,

- To set custom time range display on Scheduler
- To set different working hours
- To set different working days

Copyright © 2001 -2024 Syncfusion Inc.

- To set different first day of week
- To show/hide weekend days
- To show the week number

Set working days

By default, Scheduler considers the week days from Monday to Friday as Working days and therefore defaults to [1,2,3,4,5] - where 1 represents Monday, 2 represents Tuesday and so on. The days which are not defined in this working days collection are considered as non-working days. Therefore, when the weekend days are set to hide from Scheduler, all those non-working days too gets hidden from the layout.

The Work week and Timeline Work week views displays exactly the defined working days on Scheduler layout, whereas other views displays all the days and simply differentiates the non-working days on UI with inactive cell color.

The working or business hours depiction on Scheduler are usually valid only on these specified working days.

The following example code depicts how to set the Scheduler to display Monday, Wednesday and Friday as working days of a week.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule height='550px' :selectedDate='selectedDate'
:eventSettings='eventSettings' :currentView='currentView'
:workDays='workDays'>
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='TimelineWeek'></e-view>
                        <e-view option='TimelineWorkWeek'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Week, WorkWeek, Month, TimelineViews } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                currentView: 'WorkWeek',
                workDays: [1, 3, 5],
                eventSettings: { dataSource: scheduleData }
```

```
            }
        },
        provide: {
            schedule: [Week, WorkWeek, Month, TimelineViews]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/working-days-cs5" %}

Hiding weekend days

The `showWeekend` property is used to either show or hide the weekend days of a week and it is not applicable on Work week view (as non-working days are usually not displayed on work week view). By default, it is set to `true`. The days which are not a part of the working days collection of a Scheduler are usually considered as non-working or weekend days.

Here, the working days are defined as [1, 3, 4, 5] on Scheduler and therefore the remaining days (0, 2, 6 – Sunday, Tuesday and Saturday) are considered as non-working or weekend days and will be hidden from all the views when `showWeekend` property is set to `false`.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:workDays='workDays' :showWeekend='showWeekend'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='TimelineMonth'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, TimelineMonth } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                showWeekend: false,
                workDays: [1, 3, 4, 5],
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Day, Week, TimelineMonth]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/working-days-cs6" %}

Show week numbers

It is possible to show the week number count of a week in the header bar of the Scheduler by setting true to `showWeekNumber` property. By default, its default value is `false`. In Month view, the week numbers are displayed as a first column.

The `showWeekNumber` property is not applicable on Timeline views, as it has the equivalent headerRows property to handle such requirement with additional customizations.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
```

```
                <ejs-schedule height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:currentView='currentView' :showWeekNumber='showWeekNumber'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, Month } from '@syncfusion/ej2-vue-
schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                currentView: 'Month',
                showWeekNumber: true,
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Day, Week, Month]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/working-days-cs7" %}

Different options in showing week numbers

By default, week numbers are shown in the Scheduler based on the first day of the year. However, the week numbers can be determined based on the following criteria.

Copyright © 2001 -2024 Syncfusion Inc.

FirstDay – The first week of the year is calculated based on the first day of the year.

FirstFourDayWeek – The first week of the year begins from the first week with four or more days.

FirstFullWeek – The first week of the year begins when meeting the first day of the week (firstDayOfWeek) and the first day of the year.

For more details refer to this link

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:currentView='currentView' :showWeekNumber='showWeekNumber'
:weekRule='weekRule'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, Month } from '@syncfusion/ej2-vue-
schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                currentView: 'Month',
                showWeekNumber: true,
                weekRule: 'FirstFourDayWeek',
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Day, Week, Month]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/working-days-cs8" %}

Note: Enable the showWeekNumber property to configure the weekRule property. Also, the weekRule property depends on the value of the firstDayOfWeek property.

Set working hours

Working hours indicates the work hour limit within the Scheduler, which is visually highlighted with an active color on work cells. The working hours can be set on Scheduler using the workHours property which is of object type and includes the following sub-options,

- highlight – enables/disables the highlighting of work hours.
- start - sets the start time of the working/business hour of a day.
- end - sets the end time limit of the working/business hour of a day.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:workHours='workHours'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, WorkWeek } from '@syncfusion/ej2-
vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                workHours: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    highlight: true,
                    start: '11:00',
                    end: '20:00'
                },
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/working-days-cs9" %}

Scheduler displaying custom hours

It is possible to display the event Scheduler layout with specific time durations by hiding the unwanted hours. To do so, set the start and end hour for the Scheduler using the `startHour` and `endHour` properties respectively.

The following code example displays the Scheduler starting from the time range 7.00 AM to 6.00 PM and the remaining hours are hidden on the UI.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
startHour='07:00' endHour='18:00'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, WorkWeek } from '@syncfusion/ej2-
vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/working-days-cs10" %}

Setting start day of the week

By default, Scheduler defaults to Sunday as its first day of a week. To change the Scheduler's start day of a week with different day, set the firstDayOfWeek property with the values ranging from 0 to 6.

Here, Sunday is always denoted as 0, Monday as 1 and so on.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:firstDayOfWeek='firstDayOfWeek'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <e-views>
                        <e-view option='Week'></e-view>
                        <e-view option='Month'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Week, Month } from '@syncfusion/ej2-vue-
schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                firstDayOfWeek : 3,
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Week, Month]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/working-days-cs11" %}

Scroll to specific time and date

You can manually scroll to a specific time on Scheduler by making use of the scrollTo method as depicted in the following code example.

APP.VUE

```
<template>
    <div>
```

```
        <div id='app'>
            <div>
                <ejs-timepicker width='100' :value='date' format='HH:mm'
:change='onChange'>
                </ejs-timepicker>
            </div>
            <div id='container'>
                <ejs-schedule ref='ScheduleObj' height='510px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, WorkWeek } from '@syncfusion/ej2-
vue-schedule';
    import { TimePickerPlugin } from '@syncfusion/ej2-vue-calendars';
    Vue.use(SchedulePlugin);
    Vue.use(TimePickerPlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData },
                date: new Date(2000, 0, 1, 9)
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek]
        },
        methods: {
            onChange: function(args){
                this.$refs.ScheduleObj.ej2Instances.scrollTo(args.text);
            }
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/scroll-to-cs1" %}

How to scroll to current time on initial load

There are scenarios where you may need to load the Scheduler displaying the system's current time on the currently visible view port area. In such cases, the Scheduler needs to be scrolled to a specific time based on the system's current time which is depicted in the following code example.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule ref='scheduleObj' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:created='onCreated'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='TimelineWorkWeek'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, TimelineViews } from
'@syncfusion/ej2-vue-schedule';
    import { Internationalization } from '@syncfusion/ej2-base';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Day, Week, TimelineViews]
        },
        methods: {
            onCreated: function(){
                let intl = new Internationalization();
                this.$refs.scheduleObj.scrollTo(intl.formatDate(new Date(),
{ skeleton: 'Hm' }));
            }
        }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/working-days-cs12" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

See Also

- To display the current time indicator
- To set different working hours dynamically
- To set different working hours for each resources
- To set different working days for each resources

Cell customization in Vue Schedule component

The cells of the Scheduler can be easily customized either using the cell template or `RenderCell` event.

Setting cell dimensions in all views

The height and width of the Scheduler cells can be customized either to increase or reduce its size through the `cssClass` property, which overrides the default CSS applied on cells.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule id='Schedule' width='100%' cssClass='schedule-
cell-dimension' height='550px' :selectedDate='selectedDate'
:eventSettings='eventSettings'>
                <e-views>
                    <e-view option='Day'></e-view>
                    <e-view option='Week'></e-view>
                    <e-view option='WorkWeek'></e-view>
                    <e-view option='Month'></e-view>
                </e-views>
            </ejs-schedule>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { employeeEventData } from './datasource.js';
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: { dataSource: employeeEventData },
                selectedDate: new Date(2018, 1, 15)
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        }
    }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
.schedule-cell-dimension.e-schedule .e-vertical-view .e-date-header-wrap
table col,
.schedule-cell-dimension.e-schedule .e-vertical-view .e-content-wrap table
col {
    width: 200px;
}
.schedule-cell-dimension.e-schedule .e-vertical-view .e-time-cells-wrap
table td,
.schedule-cell-dimension.e-schedule .e-vertical-view .e-work-cells {
    height: 100px;
}
.schedule-cell-dimension.e-schedule .e-month-view .e-work-cells,
.schedule-cell-dimension.e-schedule .e-month-view .e-date-header-wrap table
col {
    width: 200px;
}
.schedule-cell-dimension.e-schedule .e-month-view .e-work-cells {
    height: 200px;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/cell-dimension-cs1" %}

Check for cell availability

You can check whether the given time range slots are available for event creation or already occupied by other events using the isSlotAvailable method. In the following code example, if a specific time slot already contains an appointment, then no more appointments can be added to that cell.

Note: The **isSlotAvailable** is centered around verifying appointments within the present view's date range. Yet, it does not encompass an evaluation of availability for recurrence occurrences that fall beyond this particular date range.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule ref='scheduleObj' width='100%' height='550px'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:actionBegin='onActionBegin'>
                <e-views>
                    <e-view option='Week'></e-view>
                    <e-view option='WorkWeek'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Week, WorkWeek, View } from '@syncfusion/ej2-
vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15)
            }
        },
        methods: {
            onActionBegin: function(args){
                if (args.requestType === 'eventCreate' && args.data.length >
0) {
                    let eventData = args.data[0];
                    let scheduleObj = this.$refs.scheduleObj.ej2Instances;
                    let eventField = scheduleObj.eventFields;
                    let startDate = eventData[eventField.startTime];
                    let endDate = eventData[eventField.endTime];
                    args.cancel = !scheduleObj.isSlotAvailable(startDate,
endDate);
                }
            }
        },
        provide: {
            schedule: [Week, WorkWeek]
        }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/cell-dimension-cs2" %}

Customizing cells in all the views

It is possible to customize the appearance of the cells using both template options and renderCell event on all the views.

Using template

The cellTemplate option accepts the template string and is used to customize the cell background with specific images or appropriate text on the given date values.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule ref='scheduleObj' width='100%' height='550px'
:selectedDate='selectedDate' :cellTemplate='cellTemplate'
cssClass='schedule-cell-template'>
                <e-views>
                    <e-view option='Day'></e-view>
                    <e-view option='Week'></e-view>
                    <e-view option='Month'></e-view>
                    <e-view option='TimelineWeek'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SchedulePlugin, Day, Week, TimelineViews, Month, View } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    var cellTemplateVue = Vue.component("cellTemplate", {
        template: `<div class="templatewrap"><div v-
if="data.type==='workCells'"><div v-
html=getWorkCellText(data.date)></div></div><div v-else-
```

```
if="data.type==='monthCells'"><div v-
html=getMonthCellText(data.date)></div></div></div>`,
        data() {
            return {
                data: {}
            };
        },
        methods: {
            getWorkCellText: function(date) {
                let weekEnds = [0, 6];
                if (weekEnds.indexOf(date.getDay()) >= 0) {
                    return "<img
src='https://ej2.syncfusion.com/demos/src/schedule/images/newyear.svg' />";
                }
                return '';
            },
            getMonthCellText: function(date) {
                if (date.getMonth() === 10 && date.getDate() === 23) {
                    return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/birthday.svg"/>';
                } else if (date.getMonth() === 11 && date.getDate() === 9) {
                    return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/get-together.svg" />';
                } else if (date.getMonth() === 11 && date.getDate() === 13)
{
                    return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/birthday.svg" />';
                } else if (date.getMonth() === 11 && date.getDate() === 22)
{
                    return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/thanksgiving-day.svg"
/>';
                } else if (date.getMonth() === 11 && date.getDate() === 24)
{
                    return '<img
src="https://ej2.syncfusion.com/demos/src/schedule/images/christmas-eve.svg"
/>';
                } else if (date.getMonth() === 11 && date.getDate() === 25)
{
                    return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/christmas.svg" />';
                } else if (date.getMonth() === 0 && date.getDate() === 1) {
                    return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/newyear.svg" />';
                } else if (date.getMonth() === 0 && date.getDate() === 14) {
                    return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/birthday.svg" />';
                }
                return '';
            }
        }
    });
    export default {
        data () {
            return {
                selectedDate: new Date(2017, 11, 16),
                cellTemplate: function (e) {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    return {
                        template: cellTemplateVue
                    };
                },
            }
        },
        provide: {
            schedule: [Day, Week, Month, TimelineViews]
        }
    }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
.templatewrap {
  text-align: center;
  /* In MONTH view the cell template is a SIBLING of event templates. So
which is necessary to set the parent position relative and the child
position absolute with 100% width */
  position: absolute;
  width: 100%;
}
.schedule-cell-template.e-schedule .e-month-view .e-work-cells {
  position: relative;
}
.templatewrap img {
  width: 20px;
  height: 20px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/cell-dimension-cs3" %}

Using renderCell event

An alternative to cellTemplate is the renderCell event, which can also be used to customize the cells with appropriate images or formatted text values.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule ref='scheduleObj' width='100%' height='550px'
:selectedDate='selectedDate' :currentView='currentView'
:renderCell='onRenderCell'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-views>
                    <e-view option='Day'></e-view>
                    <e-view option='Week'></e-view>
                    <e-view option='Month'></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { createElement } from '@syncfusion/ej2-base';
    import { SchedulePlugin, Day, Week, Month } from '@syncfusion/ej2-vue-
schedule';
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 14),
                eventSettings: { dataSource: scheduleData },
                currentView: 'Month'
            }
        },
        methods: {
            onRenderCell: (args) => {
                if (args.elementType == 'workCells' || args.elementType ==
'monthCells') {
                    let weekEnds = [0, 6];
                    if (weekEnds.indexOf((args.date).getDay()) >= 0) {
                        let ele = createElement('div', {
                            innerHTML: "<img
src='https://ej2.syncfusion.com/demos/src/schedule/images/newyear.svg' />",
                            className: 'templatewrap'
                        });
                        (args.element).appendChild(ele);
                    }
                }
            }
        },
        provide: {
            schedule: [Day, Week, Month]
        }
    }
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
.templatewrap {
  text-align: center;
  width: 100%;
}
.templatewrap img {
  width: 20px;
  height: 20px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/cell-dimension-cs4" %}

You can customize cells such as work cells, month cells, all-day cells, header cells, resource header cells using renderCell event by checking the elementType option within the event. You can check elementType with any of the following.

| Element type | Description |

|-------|---------|

| dateHeader | triggers on header cell rendering.|

| monthDay | triggers on header cell in month view rendering.|

| resourceHeader | triggers on resource header cell rendering.|

| alldayCells | triggers on all day cell rendering.|

| emptyCells | triggers on empty cell rendering on header bar.|

| resourceGroupCells | triggers on rendering of work cells for parent resource.|

| workCells | triggers on work cell rendering.|

| monthCells | triggers on month cell rendering.|

| majorSlot | triggers on major time slot cell rendering.|

| minorSlot | triggers on minor time slot cell rendering.|

| weekNumberCell | triggers on cell displaying week number.|

Customizing cell header in month view

The month header of each date cell in the month view can be customized using the cellHeaderTemplate option which accepts the string or HTMLElement. The corresponding date can be accessed with the template.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <ejs-schedule ref='scheduleObj' width='100%' height='550px'
:cellHeaderTemplate='cellHeaderTemplate' cssClass='schedule-cell-header-
template'>
                <e-views>
                    <e-view option='Month'></e-view>
                </e-views>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { Internationalization } from '@syncfusion/ej2-base';
    import { SchedulePlugin, Month } from '@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    var instance = new Internationalization();
    var cellHeaderTemplateVue = Vue.component("cellHeaderTemplate", {
        template: `<div v-html=getDateHeaderText(data.date)></div>`,
        data() {
            return {
                data: {}
            };
        },
        methods: {
            getDateHeaderText: function(date) {
                return instance.formatDate(date, { skeleton: "Ed" });
            }
        }
    });
    export default {
        data () {
            return {
                cellHeaderTemplate: function (e) {
                    return {
                        template: cellHeaderTemplateVue
                    };
                },
            }
        },
        provide: {
            schedule: [Month]
        }
    }
</script>
```

{% previewsample "page.domainurl/code-snippet/schedule/cell-dimension-cs5" %}

Customizing the minimum and maximum date values

Providing the minDate and maxDate property with some date values, allows the Scheduler to set the minimum and maximum date range. The Scheduler date that lies beyond this minimum and maximum date range will be in a disabled state so that the date navigation will be blocked beyond the specified date range.

APP.VUE

```
<template>
    <div id='app'>
        <ejs-schedule height='550px' :selectedDate='selectedDate'
:minDate='minDate' :maxDate='maxDate' :currentView='currentView'>
        </ejs-schedule>
    </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
  import Vue from 'vue';
  import { scheduleData } from './datasource.js';
  import { SchedulePlugin, Day, WorkWeek, Agenda, Month, Week } from
'@syncfusion/ej2-vue-schedule';
  Vue.use(SchedulePlugin);
  export default {
    data () {
      return {
        selectedDate: new Date(2018, 1, 17),
        minDate: new Date(2017, 4, 17),
        maxDate: new Date(2018, 5, 17)
        eventSettings: { dataSource: scheduleData },
        currentView: 'Month',
      }
    },
    provide: {
      schedule: [Day, WorkWeek, Agenda, Month, Week]
    }
  }
</script>
```

{% previewsample "page.domainurl/code-snippet/schedule/cell-dimension-cs6" %}

By default, the minDate property value is set to new Date(1900, 0, 1) and maxDate property value is set to new Date(2099, 11, 31). The user can also set the customized minDate and maxDate property values.

How to disable multiple cell and row selection in Schedule

By default, the allowMultiCellSelection and allowMultiRowSelection properties of the Schedule are set to true. So, the Schedule allows user to select multiple cells and rows. If the user want to disable this multiple cell and row selection. The user can disable this feature by setting up false to these properties.

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

State persistence in Vue Schedule component

State persistence allowed Scheduler to retain the currentView, selectedDate and Scroll position values in the localStorage for state maintenance even if the browser is refreshed or if you move to the next page within the browser. This action is handled through the enablePersistence property which is set to false by default. When it is set to true, currentView, selectedDate and Scroll position values of the scheduler component will be retained even after refreshing the page.

Note: Scheduler id is essential to set state persistence.

The following sample demonstrates how to set state persistence of the Scheduler component.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
```

```
                <ejs-schedule id='schedule' height='550px' width='100%'
:selectedDate='selectedDate' :eventSettings='eventSettings'
:enablePersistence='enablePersistence'>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import {SchedulePlugin, Day, Week, TimelineViews, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                selectedDate: new Date(2018, 1, 15),
                views: ['Day', 'Week', 'TimelineWeek', 'Month', 'Agenda'],
                enablePersistence: true,
                eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
           schedule: [Day, Week, TimelineViews, Month, Agenda]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/working-days-cs4" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Exporting in Vue Schedule component

The Scheduler supports exporting all its appointments both to an Excel or ICS extension file at client-side. It offers different client-side methods to export its appointments in an Excel or ICal format file. Let's look onto the ways on how to implement the exporting functionality in Scheduler.

Copyright © 2001 -2024 Syncfusion Inc.

Excel Exporting

The Scheduler allows you to export all its events into an Excel format file by using the [exportToExcel] client-side method. By default, it exports all the default fields of Scheduler mapped through eventSettings property.

Before you start with excel exporting functionality, you need to import and inject the ExcelExport module from the '@syncfusion/ej2-schedule' package using the Inject method of Scheduler.

APP.VUE

```
<template>
  <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' ref="ScheduleObj"
:cssClass="cssClass" height="550px" :selectedDate='selectedDate'
:eventSettings='eventSettings'
                    :currentView="currentView" :actionBegin="onActionBegin">
                    <e-views>
                        <e-view option="Week"></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
</template>
<script>
import Vue from "vue";
import { SchedulePlugin, Week, View, Resize, DragAndDrop, ExcelExport
} from "@syncfusion/ej2-vue-schedule";
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
        data: function () {
            return {
                cssClass: 'excel-export',
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2019, 0, 10),
                currentView: 'Week'
            }
        },
        provide: {
            schedule: [Week, Resize, DragAndDrop, ExcelExport]
        },
        methods: {
            onActionBegin: function (args) {
                if (args.requestType === 'toolbarItemRendering') {
                    let exportItem = {
                        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-
icon-schedule-excel-export',
                        text: 'Excel Export', cssClass: 'e-excel-export',
click: this.onExportClick.bind(this)
                    };
                    args.items.push(exportItem);
                }
            },
            onExportClick: function () {
                let scheduleObj = this.$refs.ScheduleObj;
```

```
                scheduleObj.exportToExcel();
            }
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
    .excel-export.e-schedule .e-schedule-toolbar .e-icon-schedule-excel-
export::before {
        content: '\e242';
    }
    .excel-export.e-schedule .e-schedule-toolbar .e-toolbar-item.e-today{
        display: none !important;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/excel-export-cs1" %}

Exporting with custom fields

By default, Scheduler exports all the default event fields that are mapped to it through the eventSettings property. To limit the number of fields on the exported excel file, it provides an option to export only the custom fields of the event data. To export such custom fields alone, define the required fields through the ExportOptions interface and pass it as argument to the exportToExcel method as shown in the following example. For example: ['Id', 'Subject', 'StartTime', 'EndTime', 'Location'].

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' ref="ScheduleObj"
:cssClass="cssClass" height="550px" :selectedDate='selectedDate'
:eventSettings='eventSettings'
                    :currentView="currentView" :actionBegin="onActionBegin">
                    <e-views>
                        <e-view option="Week"></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { SchedulePlugin, Week, View, Resize, DragAndDrop, ExcelExport
} from "@syncfusion/ej2-vue-schedule";
import { scheduleData } from './datasource.js';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(SchedulePlugin);
export default {
        data: function () {
            return {
                cssClass: 'excel-export',
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2019, 0, 10),
                currentView: 'Week'
            }
        },
        provide: {
            schedule: [Week, Resize, DragAndDrop, ExcelExport]
        },
        methods: {
            onActionBegin: function (args) {
                if (args.requestType === 'toolbarItemRendering') {
                    let exportItem = {
                        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-
icon-schedule-excel-export',
                        text: 'Excel Export', cssClass: 'e-excel-export',
click: this.onExportClick.bind(this)
                    };
                    args.items.push(exportItem);
                }
            },
            onExportClick: function () {
                let scheduleObj = this.$refs.ScheduleObj;
                let exportValues = {
                    fields: ['Id', 'Subject', 'StartTime', 'EndTime',
'Location']
                };
                scheduleObj.exportToExcel(exportValues);
            }
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
    .excel-export.e-schedule .e-schedule-toolbar .e-icon-schedule-excel-
export::before {
        content: '\e242';
    }
    .excel-export.e-schedule .e-schedule-toolbar .e-toolbar-item.e-today{
        display: none !important;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/excel-export-cs2" %}

Copyright © 2001 -2024 Syncfusion Inc.

Exporting individual occurrences of a recurring series

By default, the Scheduler exports recurring events as a single data by exporting only its parent record into the excel file. If you want to export each individual occurrences of a recurring series appointment as separate records in an Excel file, define the includeOccurrences option as true through the ExportOptions interface and pass it as argument to the exportToExcel method. By default, the includeOccurrences option is set to false.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' ref="ScheduleObj"
:cssClass="cssClass" height="550px" :selectedDate='selectedDate'
:eventSettings='eventSettings'
                    :currentView="currentView" :actionBegin="onActionBegin">
                    <e-views>
                        <e-view option="Week"></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { SchedulePlugin, Week, View, Resize, DragAndDrop, ExcelExport
} from "@syncfusion/ej2-vue-schedule";
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
        data: function () {
            return {
                cssClass: 'excel-export',
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2019, 0, 10),
                currentView: 'Week'
            }
        },
        provide: {
            schedule: [Week, Resize, DragAndDrop, ExcelExport]
        },
        methods: {
            onActionBegin: function (args) {
                if (args.requestType === 'toolbarItemRendering') {
                    let exportItem = {
                        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-
icon-schedule-excel-export',
                        text: 'Excel Export', cssClass: 'e-excel-export',
click: this.onExportClick.bind(this)
                    };
                    args.items.push(exportItem);
                }
            },
            onExportClick: function () {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                let scheduleObj = this.$refs.ScheduleObj;
                let exportValues = { includeOccurrences: true };
                scheduleObj.exportToExcel(exportValues);
            }
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
    .excel-export.e-schedule .e-schedule-toolbar .e-icon-schedule-excel-
export::before {
        content: '\e242';
    }
    .excel-export.e-schedule .e-schedule-toolbar .e-toolbar-item.e-today{
        display: none !important;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/excel-export-cs3" %}

Exporting custom event data

By default, the whole event collection bound to the Scheduler gets exported as an excel file. To export only specific events of Scheduler or some custom event collection, you need to pass those custom data collection as a parameter to the exportToExcel method as shown in this following example, through the customData option of ExportOptions interface.

By default, the event data are taken from Scheduler dataSource.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' ref="ScheduleObj"
:cssClass="cssClass" height="550px" :selectedDate='selectedDate'
:eventSettings='eventSettings'
                    :currentView="currentView" :actionBegin="onActionBegin">
                    <e-views>
                        <e-view option="Week"></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { SchedulePlugin, Week, View, Resize, DragAndDrop, ExcelExport
```

```
} from "@syncfusion/ej2-vue-schedule";
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
        data: function () {
            return {
                cssClass: 'excel-export',
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2019, 0, 10),
                currentView: 'Week'
            }
        },
        provide: {
            schedule: [Week, Resize, DragAndDrop, ExcelExport]
        },
        methods: {
            onActionBegin: function (args) {
                if (args.requestType === 'toolbarItemRendering') {
                    let exportItem = {
                        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-
icon-schedule-excel-export',
                        text: 'Excel Export', cssClass: 'e-excel-export',
click: this.onExportClick.bind(this)
                    };
                    args.items.push(exportItem);
                }
            },
            onExportClick: function () {
                let scheduleObj = this.$refs.ScheduleObj;
                let exportValues = {
                    customData: [{
                        Id: 1,
                        Subject: 'Explosion of Betelgeuse Star',
                        Location: 'Space Centre USA',
                        StartTime: new Date(2019, 0, 6, 9, 30),
                        EndTime: new Date(2019, 0, 6, 11, 0),
                        CategoryColor: '#1aaa55'
                    }, {
                        Id: 2,
                        Subject: 'Thule Air Crash Report',
                        Location: 'Newyork City',
                        StartTime: new Date(2019, 0, 7, 12, 0),
                        EndTime: new Date(2019, 0, 7, 14, 0),
                        CategoryColor: '#357cd2'
                    }]
                };
                scheduleObj.exportToExcel(exportValues);
            }
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
    .excel-export.e-schedule .e-schedule-toolbar .e-icon-schedule-excel-
export::before {
        content: '\e242';
    }
    .excel-export.e-schedule .e-schedule-toolbar .e-toolbar-item.e-today{
        display: none !important;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/excel-export-cs4" %}

Customizing column header with custom fields exporting

Using fields property, we can only export the defined fields into excel without customizing the header. Now we can provide the alternate support to customize the header of custom fields exporting using the fieldsInfo option through the ExportFieldInfo interface and pass it as an argument to the exportToExcel method as shown in the following example.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' ref="ScheduleObj"
:cssClass="cssClass" height="550px" :selectedDate='selectedDate'
:eventSettings='eventSettings'
                    :currentView="currentView" :actionBegin="onActionBegin">
                    <e-views>
                        <e-view option="Week"></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { SchedulePlugin, Week, View, Resize, DragAndDrop, ExcelExport
} from "@syncfusion/ej2-vue-schedule";
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
        data: function () {
            return {
                cssClass: 'excel-export',
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2019, 0, 10),
                currentView: 'Week'
            }
        },
        provide: {
            schedule: [Week, Resize, DragAndDrop, ExcelExport]
        },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        methods: {
            onActionBegin: function (args) {
                if (args.requestType === 'toolbarItemRendering') {
                    let exportItem = {
                        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-
icon-schedule-excel-export',
                        text: 'Excel Export', cssClass: 'e-excel-export',
click: this.onExportClick.bind(this)
                    };
                    args.items.push(exportItem);
                }
            },
            onExportClick: function () {
                let scheduleObj = this.$refs.ScheduleObj;
                let customFields = [
                    { name: 'Subject', text: 'Summary' },
                    { name: 'StartTime', text: 'First Date' },
                    { name: 'EndTime', text: 'Last Date' },
                    { name: 'Location', text: 'Place' },
                    { name: 'OwnerId', text: 'Owners' }
                ];
                let exportValues = { fieldsInfo: customFields };
                scheduleObj.exportToExcel(exportValues);
            }
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
    .excel-export.e-schedule .e-schedule-toolbar .e-icon-schedule-excel-
export::before {
        content: '\e242';
    }
    .excel-export.e-schedule .e-schedule-toolbar .e-toolbar-item.e-today{
        display: none !important;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/excel-export-cs5" %}

Export with custom file name

By default, the Scheduler allows you to download the exported Excel file with a name Schedule.xlsx. It also provides an option to export the excel file with a custom file name, by defining the desired fileName through the ExportOptions interface and passing it as an argument to the exportToExcel method.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' ref="ScheduleObj"
:cssClass="cssClass" height="550px" :selectedDate='selectedDate'
:eventSettings='eventSettings'
                    :currentView="currentView" :actionBegin="onActionBegin">
                    <e-views>
                        <e-view option="Week"></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { SchedulePlugin, Week, View, Resize, DragAndDrop, ExcelExport
} from "@syncfusion/ej2-vue-schedule";
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
        data: function () {
            return {
                cssClass: 'excel-export',
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2019, 0, 10),
                currentView: 'Week'
            }
        },
        provide: {
            schedule: [Week, Resize, DragAndDrop, ExcelExport]
        },
        methods: {
            onActionBegin: function (args) {
                if (args.requestType === 'toolbarItemRendering') {
                    let exportItem = {
                        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-
icon-schedule-excel-export',
                        text: 'Excel Export', cssClass: 'e-excel-export',
click: this.onExportClick.bind(this)
                    };
                    args.items.push(exportItem);
                }
            },
            onExportClick: function () {
                let scheduleObj = this.$refs.ScheduleObj;
                let exportValues = { fileName: "SchedulerData" };
                scheduleObj.exportToExcel(exportValues);
            }
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
    .excel-export.e-schedule .e-schedule-toolbar .e-icon-schedule-excel-
export::before {
        content: '\e242';
    }
    .excel-export.e-schedule .e-schedule-toolbar .e-toolbar-item.e-today{
        display: none !important;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/excel-export-cs6" %}

Excel file formats

By default, the Scheduler exports event data to an excel file in the .xlsx format. You can also export the Scheduler data in either of the file type such as .xlsx or csv formats, by defining the exportType option as either csv or xlsx through the ExportOptions interface. By default, the exportType is set to xlsx.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' ref="ScheduleObj"
:cssClass="cssClass" height="550px" :selectedDate='selectedDate'
:eventSettings='eventSettings'
                    :currentView="currentView" :actionBegin="onActionBegin">
                    <e-views>
                        <e-view option="Week"></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { SchedulePlugin, Week, View, Resize, DragAndDrop, ExcelExport
} from "@syncfusion/ej2-vue-schedule";
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
        data: function () {
            return {
                cssClass: 'excel-export',
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2019, 0, 10),
                currentView: 'Week'
            }
        },
        provide: {
```

```
            schedule: [Week, Resize, DragAndDrop, ExcelExport]
        },
        methods: {
            onActionBegin: function (args) {
                if (args.requestType === 'toolbarItemRendering') {
                    let exportItem = {
                        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-
icon-schedule-excel-export',
                        text: 'Excel Export', cssClass: 'e-excel-export',
click: this.onExportClick.bind(this)
                    };
                    args.items.push(exportItem);
                }
            },
            onExportClick: function () {
                let scheduleObj = this.$refs.ScheduleObj;
                let exportValues = { exportType: "csv" };
                scheduleObj.exportToExcel(exportValues);
            }
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
    .excel-export.e-schedule .e-schedule-toolbar .e-icon-schedule-excel-
export::before {
        content: '\e242';
    }
    .excel-export.e-schedule .e-schedule-toolbar .e-toolbar-item.e-today{
        display: none !important;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/excel-export-cs7" %}

Custom separator in CSV

The Scheduler exports the event data to CSV format with , as separator. You can change separator by setting separator property in ExportOptions.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' ref="ScheduleObj"
:cssClass="cssClass" height="550px" :selectedDate='selectedDate'
:eventSettings='eventSettings'
                    :currentView="currentView" :actionBegin="onActionBegin">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <e-views>
                        <e-view option="Week"></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { SchedulePlugin, Week, View, Resize, DragAndDrop, ExcelExport
} from "@syncfusion/ej2-vue-schedule";
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
        data: function () {
            return {
                cssClass: 'excel-export',
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2019, 0, 10),
                currentView: 'Week'
            }
        },
        provide: {
            schedule: [Week, Resize, DragAndDrop, ExcelExport]
        },
        methods: {
            onActionBegin: function (args) {
                if (args.requestType === 'toolbarItemRendering') {
                    let exportItem = {
                        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-
icon-schedule-excel-export',
                        text: 'CSV-Export', cssClass: 'e-excel-export',
click: this.onExportClick.bind(this)
                    };
                    args.items.push(exportItem);
                }
            },
            onExportClick: function () {
                let scheduleObj = this.$refs.ScheduleObj;
                let exportValues = { exportType: 'csv', separator: ';' };
                scheduleObj.exportToExcel(exportValues);
            }
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
    .excel-export.e-schedule .e-schedule-toolbar .e-icon-schedule-excel-
export::before {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        content: '\e242';
    }
    .excel-export.e-schedule .e-schedule-toolbar .e-toolbar-item.e-today{
        display: none !important;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/excel-export-cs8" %}

Exporting calendar events as ICS file

You can export the Scheduler events to a calendar (.ics) file format, and open it on any of the other default calendars such as Google or Outlook. To export the events of Scheduler to an ICS file, you need to first import the ICalendarExport module from @syncfusion/ej2-schedule package and then inject it using the Schedule.Inject(ICalendarExport) method.

The following code example shows how the Scheduler events are exported to a calendar (.ics) file by making use of the exportToICalendar public method.

APP.VUE

```
<template>
  <div>
        <div id='app'>
            <div id='container'>
                <ejs-button id='ics-export' v-
on:click.native="onClick">Export</ejs-button>
                <ejs-schedule id="Schedule" ref="ScheduleObj" height="520px"
:selectedDate='selectedDate'
                    :eventSettings='eventSettings'
:currentView='currentView'>
                    <e-views>
                        <e-view option="Day"></e-view>
                        <e-view option="Week"></e-view>
                        <e-view option="WorkWeek"></e-view>
                        <e-view option="Month"></e-view>
                        <e-view option="Agenda"></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, Resize,
DragAndDrop, ICalendarExport, ICalendarImport } from "@syncfusion/ej2-vue-
schedule";
    import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
    import { scheduleData } from './datasource.js';
    Vue.use(ButtonPlugin);
    Vue.use(SchedulePlugin);
    export default {
        data: function () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15),
```

```
                currentView: 'Week',
                cssClass:'calendar-import',
                multiple: false,
                allowedExtensions: '.ics',
                buttons: {
                    browse: 'Choose file',
                },
                showFileList: false,
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda, Resize,
DragAndDrop, ICalendarExport, ICalendarImport]
        },
        methods: {
            onClick: function () {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                scheduleObj.exportToICalendar();
            }
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
    .calendar-import.e-upload {
        border: 0;
        padding-left: 0 !important;
    }
    .calendar-import.e-upload .e-file-select-wrap {
        padding: 0
    }
    .calendar-import.e-upload .e-file-select-wrap .e-file-drop {
        display: none;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/calendar-export-cs1" %}

Exporting calendar with custom file name

By default, the calendar is exported with a file name `Calendar.ics`. To change this file name on export, pass the custom string value as `fileName` to the method argument so as to get the file downloaded with this provided name.

The following example downloads the iCal file with a name `ScheduleEvents.ics`.

APP.VUE

```
<template>
```

```
  <div>
        <div id='app'>
            <div id='container'>
                <ejs-button id='ics-export' v-
on:click.native="onClick">Export</ejs-button>
                <ejs-schedule id="Schedule" ref="ScheduleObj" height="520px"
:selectedDate='selectedDate'
                    :eventSettings='eventSettings'
:currentView='currentView'>
                    <e-views>
                        <e-view option="Day"></e-view>
                        <e-view option="Week"></e-view>
                        <e-view option="WorkWeek"></e-view>
                        <e-view option="Month"></e-view>
                        <e-view option="Agenda"></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, Resize,
DragAndDrop, ICalendarExport, ICalendarImport } from "@syncfusion/ej2-vue-
schedule";
    import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
    import { scheduleData } from './datasource.js';
    Vue.use(ButtonPlugin);
    Vue.use(SchedulePlugin);
    export default {
        data: function () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15),
                currentView: 'Week',
                cssClass:'calendar-import',
                multiple: false,
                allowedExtensions: '.ics',
                buttons: {
                    browse: 'Choose file',
                },
                showFileList: false,
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda, Resize,
DragAndDrop, ICalendarExport, ICalendarImport]
        },
        methods: {
            onClick: function () {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                scheduleObj.exportToICalendar('ScheduleEvents');
            }
        }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
    .calendar-import.e-upload {
        border: 0;
        padding-left: 0 !important;
    }
    .calendar-import.e-upload .e-file-select-wrap {
        padding: 0
    }
    .calendar-import.e-upload .e-file-select-wrap .e-file-drop {
        display: none;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/calendar-export-cs2" %}

Import events from other calendars

The events from external calendars (ICS files) can be imported into Scheduler by using the importICalendar method. This method accepts the blob object of an .ics file to be imported, as a mandatory argument.

To import an ICS file containing events into Scheduler, you need to first import the ICalendarImport module from @syncfusion/ej2-schedule package and then inject it using the Schedule.Inject(ICalendarImport) method.

The following example shows how to import an ICS file into Scheduler, using the importICalendar method.

APP.VUE

```
<template>
  <div>
        <div id='app'>
            <div id='container'>
                <ejs-uploader id='ics-import' :cssClass='cssClass'
name="ics-import" :buttons="buttons"
                :showFileList='showFileList' :multiple='multiple'
:allowedExtensions='extensions'
                :selected='onSelect'/>
                <ejs-schedule id="Schedule" ref="ScheduleObj" height="520px"
:selectedDate='selectedDate'
                    :eventSettings='eventSettings'
:currentView='currentView'>
                    <e-views>
                        <e-view option="Day"></e-view>
                        <e-view option="Week"></e-view>
                        <e-view option="WorkWeek"></e-view>
```

```
                        <e-view option="Month"></e-view>
                        <e-view option="Agenda"></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, Resize,
DragAndDrop, ICalendarExport, ICalendarImport } from "@syncfusion/ej2-vue-
schedule";
    import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
    import { scheduleData } from './datasource.js';
    Vue.use(UploaderPlugin);
    Vue.use(SchedulePlugin);
    export default {
        data: function () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15),
                currentView: 'Week',
                cssClass:'calendar-import',
                multiple: false,
                extensions: '.ics',
                buttons: {
                    browse: 'Choose file',
                },
                showFileList: false,
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda, Resize,
DragAndDrop, ICalendarExport, ICalendarImport]
        },
        methods: {
            onSelect: function (args) {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                scheduleObj.importICalendar(args.event.target.files[0]);
            }
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
    .calendar-import.e-upload {
        border: 0;
        padding-left: 0 !important;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
    .calendar-import.e-upload .e-file-select-wrap {
        padding: 0
    }
    .calendar-import.e-upload .e-file-select-wrap .e-file-drop {
        display: none;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/calendar-import-cs1" %}

How to print the Scheduler element

The Scheduler allows you to print the Scheduler element by using the print client-side method. The print method works in two ways. You can find it below.

- Using print method without options.
- Using a print method with options.

To print the Schedule, you need to import the `Print` module from the `@syncfusion/ej2-vue-schedule` package and then inject it using `provide: {schedule: [Print]}`.

Using print method without options

You can print the Schedule element with the current view by using the print method without passing the options. The following example shows how to print the Scheduler using the `print` method without passing options.

APP.VUE

```
<template>
  <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id="Schedule" ref="ScheduleObj" height="520px"
:selectedDate='selectedDate'
                    :eventSettings='eventSettings'
:actionBegin="onActionBegin">
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, Print }
from "@syncfusion/ej2-vue-schedule";
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data: function () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15),
                cssClass:'schedule-print',
            }
```

```
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda, Print]
        },
        methods: {
            onActionBegin: function (args) {
                if (args.requestType === 'toolbarItemRendering') {
                    let exportItem = {
                        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-
icon-schedule-print',
                        text: 'Print', cssClass: 'e-print', click:
this.onPrintIconClick.bind(this)
                    };
                    args.items.push(exportItem);
                }
            },
            onPrintIconClick: function () {
                let scheduleObj = this.$refs.ScheduleObj;
                scheduleObj.print();
            }
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
.e-schedule .e-schedule-toolbar .e-icon-schedule-print::before {
  content: '\e813';
}
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/print-cs1" %}

Using a print method with options

You can print the Schedule element based on your needs using the print method by passing the print options used in this example with its values. The following example shows how to print the Scheduler using the print method by passing the options.

APP.VUE

```
<template>
  <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id="Schedule" ref="ScheduleObj" height="520px"
:selectedDate='selectedDate'
                    :eventSettings='eventSettings'
:actionBegin="onActionBegin">
                </ejs-schedule>
            </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, Print }
from "@syncfusion/ej2-vue-schedule";
    import { scheduleData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data: function () {
            return {
                eventSettings: { dataSource: scheduleData },
                selectedDate: new Date(2018, 1, 15),
                cssClass:'schedule-print',
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda, Print]
        },
        methods: {
            onActionBegin: function (args) {
                if (args.requestType === 'toolbarItemRendering') {
                    let exportItem = {
                        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-
icon-schedule-print',
                        text: 'Print', cssClass: 'e-print', click:
this.onPrintIconClick.bind(this)
                    };
                    args.items.push(exportItem);
                }
            },
            onPrintIconClick: function () {
                let scheduleObj = this.$refs.ScheduleObj;
                let printModel = {
                    agendaDaysCount: 14,
                    cssClass: 'e-print-schedule',
                    currentView: scheduleObj.currentView,
                    dateFormat: 'dd-MMM-yyyy',
                    enableRtl: false,
                    endHour: '18:00',
                    firstDayOfWeek: 1,
                    firstMonthOfYear: 0,
                    group: {},
                    height: 'auto',
                    locale: scheduleObj.locale,
                    maxDate: scheduleObj.selectedDate,
                    minDate: scheduleObj.getCurrentViewDates()[0],
                    readonly: true,
                    resources: [],
                    rowAutoHeight: false,
                    selectedDate: new Date(),
                    showHeaderBar: false,
                    showTimeIndicator: false,
                    showWeekNumber: false,
                    showWeekend: false,
                    startHour: '06:00',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    timeFormat: 'HH',
                    timeScale: { enable: true },
                    width: 'auto',
                    workDays: [1, 2, 3, 4, 5],
                    workHours: { highlight: true, start: '10:00', end:
'20:00' }
                };
                scheduleObj.print(printModel);
            }
        }
    }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-schedule/styles/material.css";
.e-schedule .e-schedule-toolbar .e-icon-schedule-print::before {
  content: '\e813';
}
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/print-cs2" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Context menu in Vue Schedule component

You can display context menu on work cells and appointments of Scheduler by making use of the ContextMenu control manually from the application end. In the following code example, context menu control is being added from sample end and set its target as Scheduler.

On Scheduler cells, you can display the menu items such as New Event, New Recurring Event and Today option. For appointments, you can display its related options such as Edit Event and Delete Event. The default event window can be opened for appointment creation and editing using the openEditor method of Scheduler.

The deletion of appointments can be done by using the deleteEvent public method. Also, the selectedDate property can be used to navigate between different dates.

You can also display custom menu options on Scheduler cells and appointments. Context menu will open on tap-hold in responsive mode.

APP.VUE

```
<template>
    <div id='app'>
        <div id='container'>
            <div class='content-wrapper'>
                <ejs-schedule id='Schedule' ref='scheduleObj' height='550px'
:selectedDate='selectedDate' :eventSettings='eventSettings'></ejs-schedule>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            </div>
            <div>
                <ejs-contextmenu id='contextmenu' ref='menuObj'
cssClass='schedule-context-menu' target='.e-schedule' :items='menuItems'
:beforeOpen='onContextMenuBeforeOpen' :select='onMenuItemSelect' ></ejs-
contextmenu>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { extend, closest, isNullOrUndefined, Browser, remove,
removeClass } from '@syncfusion/ej2-base';
    import { DataManager, Query } from '@syncfusion/ej2-data';
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
    import { ContextMenuPlugin } from '@syncfusion/ej2-vue-navigations';
    Vue.use(SchedulePlugin);
    Vue.use(ContextMenuPlugin);
    export default {
        data: function () {
            return {
                eventSettings: { dataSource: extend([], scheduleData, null,
true) },
                selectedDate: new Date(2018, 1, 15),
                menuItems: [
                    {
                        text: 'New Event',
                        iconCss: 'e-icons new',
                        id: 'Add'
                    }, {
                        text: 'New Recurring Event',
                        iconCss: 'e-icons recurrence',
                        id: 'AddRecurrence'
                    }, {
                        text: 'Today',
                        iconCss: 'e-icons today',
                        id: 'Today'
                    }, {
                        text: 'Edit Event',
                        iconCss: 'e-icons edit',
                        id: 'Save'
                    }, {
                        text: 'Edit Event',
                        id: 'EditRecurrenceEvent',
                        iconCss: 'e-icons edit',
                        items: [{
                            text: 'Edit Occurrence',
                            id: 'EditOccurrence'
                        }, {
                            text: 'Edit Series',
                            id: 'EditSeries'
                        }]
                    }, {
                        text: 'Delete Event',
```

```
                        iconCss: 'e-icons delete',
                        id: 'Delete'
                    }, {
                        text: 'Delete Event',
                        id: 'DeleteRecurrenceEvent',
                        iconCss: 'e-icons delete',
                        items: [{
                            text: 'Delete Occurrence',
                            id: 'DeleteOccurrence'
                        }, {
                            text: 'Delete Series',
                            id: 'DeleteSeries'
                        }]
                    }
                ],
                selectedTarget: null
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda]
        },
        methods: {
            onContextMenuBeforeOpen: function(args) {
                let newEventElement = document.querySelector('.e-new-
event');
                if (newEventElement) {
                    remove(newEventElement);
                    removeClass([document.querySelector('.e-selected-
cell')], 'e-selected-cell');
                }
                let scheduleObj = this.$refs.scheduleObj;
                scheduleObj.closeQuickInfoPopup();
                let targetElement = args.event.target;
                if (closest(targetElement, '.e-contextmenu')) {
                    return;
                }
                let menuObj = this.$refs.menuObj;
                this.selectedTarget = closest(targetElement, '.e-
appointment,.e-work-cells,' +
                    '.e-vertical-view .e-date-header-wrap .e-all-day-
cells,.e-vertical-view .e-date-header-wrap .e-header-cells');
                if (isNullOrUndefined(this.selectedTarget)) {
                    args.cancel = true;
                    return;
                }
                if (this.selectedTarget.classList.contains('e-appointment'))
{
                    let eventObj=
scheduleObj.getEventDetails(this.selectedTarget);
                    if (eventObj.RecurrenceRule) {
                        menuObj.showItems(['EditRecurrenceEvent',
'DeleteRecurrenceEvent'], true);
                        menuObj.hideItems(['Add', 'AddRecurrence', 'Today',
'Save', 'Delete'], true);
                    } else {
                        menuObj.showItems(['Save', 'Delete'], true);
```

```
                        menuObj.hideItems(['Add', 'AddRecurrence', 'Today',
'EditRecurrenceEvent', 'DeleteRecurrenceEvent'], true);
                    }
                    return;
                }
                menuObj.hideItems(['Save', 'Delete', 'EditRecurrenceEvent',
'DeleteRecurrenceEvent'], true);
                menuObj.showItems(['Add', 'AddRecurrence', 'Today'], true);
            },
            onMenuItemSelect: function(args) {
                let scheduleObj = this.$refs.scheduleObj.ej2Instances;
                let selectedMenuItem = args.item.id;
                let eventObj;
                if (this.selectedTarget.classList.contains('e-appointment'))
{
                    eventObj =
scheduleObj.getEventDetails(this.selectedTarget);
                }
                switch (selectedMenuItem) {
                    case 'Today':
                        scheduleObj.selectedDate = new Date();
                        break;
                    case 'Add':
                    case 'AddRecurrence':
                        let selectedCells =
scheduleObj.getSelectedElements();
                        let activeCellsData =
scheduleObj.getCellDetails(selectedCells.length > 0 ? selectedCells :
this.selectedTarget);
                        if (selectedMenuItem === 'Add') {
                            scheduleObj.openEditor(activeCellsData, 'Add');
                        } else {
                            scheduleObj.openEditor(activeCellsData, 'Add',
null, 1);
                        }
                        break;
                    case 'Save':
                    case 'EditOccurrence':
                    case 'EditSeries':
                        if (selectedMenuItem === 'EditSeries') {
                            eventObj = new
DataManager(scheduleObj.eventsData).
                                executeLocal(new
Query().where(scheduleObj.eventFields.id, 'equal',
eventObj[scheduleObj.eventFields.recurrenceID]))[0];
                        }
                        scheduleObj.openEditor(eventObj, selectedMenuItem);
                        break;
                    case 'Delete':
                        scheduleObj.deleteEvent(eventObj);
                        break;
                    case 'DeleteOccurrence':
                    case 'DeleteSeries':
                        scheduleObj.deleteEvent(eventObj, selectedMenuItem);
                        break;
                }
            }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
  .schedule-context-menu .e-menu-item .new::before {
        content: '\e7f9';
    }
    .schedule-context-menu .e-menu-item .edit::before {
        content: '\ea9a';
    }
    .schedule-context-menu .e-menu-item .recurrence::before {
        content: '\e308';
        font-weight: bold;
    }
    .schedule-context-menu .e-menu-item .today::before {
        content: '\e322';
    }
    .schedule-context-menu .e-menu-item .delete::before {
        content: '\e94a';
    }
    .e-bigger .schedule-context-menu ul .e-menu-item .e-menu-icon {
        font-size: 14px;
    }
    .schedule-context-menu ul .e-menu-item .e-menu-icon {
        font-size: 12px;
    }
    .schedule-context-menu .e-menu-parent.e-ul .e-menu-item {
        font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif",
"sans-serif", "-apple-system", "BlinkMacSystemFont";
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/context-menu-cs1" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Dimensions in Vue Schedule component

The Scheduler dimensions refers to both height and width of the entire layout and it accepts 3 types of values.

Copyright © 2001 -2024 Syncfusion Inc.

- auto
- pixel
- percentage

Auto Height and Width

When height and width of the Scheduler are set to auto, it will try as hard as possible to keep an element the same width as its parent container. In other words, the parent container that holds Scheduler, it's width/height will be the sum of its children. By default, Scheduler is assigned with auto values for both height and width properties.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: 'auto',
      width: 'auto',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/dimension-cs1" %}

Height and Width in pixel

The Scheduler height and width will be rendered exactly as per the given pixel values. It accepts both string and number values.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550',
      width: '650',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/dimension-cs2" %}

Height and Width in percentage

When height and width of the Scheduler are given as percentage, it will make the Scheduler as wide as the parent container.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate' :eventSettings='eventSettings'></ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { scheduleData } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '100%',
      width: '100%',
      eventSettings: { dataSource: scheduleData },
      selectedDate: new Date(2018, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/dimension-cs3" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

See Also

- How to Change Scheduler Cell Dimensions

Localization in Vue Schedule component

The Scheduler integrates different date-time formats and cultures, which allows it to function globally, thus meeting the diverse needs of different regions.

You can adapt the Scheduler to various languages by parsing and formatting the date or number (Internationalization), adding culture specific customization and translation to the text (Localization).

Globalization

The Internationalization library provides support for formatting and parsing the number, date, and time by using the official Unicode CLDR JSON data and also provides the `loadCldr` method to load the culture specific CLDR JSON data.

By default, Scheduler is set to follow the English culture ('en-US'). If you want to go with different culture other than English, follow the below steps.

- Install the `CLDR-Data` package by using the below command (it installs the CLDR JSON data). For more information about CLDR-Data, refer to this link.

```
npm install cldr-data --save
```

Once the package is installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

- Now import the installed CLDR JSON data into the `app.vue` file. To import JSON data, you need to install the JSON plugin loader. Here, we have used the SystemJS JSON plugin loader.

```bash
npm install systemjs-plugin-json --save-dev
```

- Once installed, configure the `system.config.js` configuration settings as shown in the following code to map the `systemjs-plugin-json` loader.

```ts
System.config({
paths: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
'syncfusion:': 'npm:@syncfusion/'
},
map: {
app: 'app',
//Syncfusion packages mapping
"@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
"@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
"@syncfusion/ej2-schedule": "syncfusion:ej2-schedule/dist/ej2-schedule.umd.min.js",
"@syncfusion/ej2-calendars": "syncfusion:ej2-calendars/dist/ej2-calendars.umd.min.js",
"@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",
"@syncfusion/ej2-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",
"@syncfusion/ej2-lists": "syncfusion:ej2-lists/dist/ej2-lists.umd.min.js",
"@syncfusion/ej2-navigations": "syncfusion:ej2-navigations/dist/ej2-navigations.umd.min.js",
"@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",
"@syncfusion/ej2-dropdowns": "syncfusion:ej2-dropdowns/dist/ej2-dropdowns.umd.min.js",
"@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js",
"cldr-data": 'npm:cldr-data',
"plugin-json": "npm:systemjs-plugin-json/json.js"
},
meta: {
'*.json': { loader: 'plugin-json' }
},
packages: {
'app': { main: 'app', defaultExtension: 'js' },
'cldr-data': { main: 'index.js', defaultExtension: 'js' }
}
});
System.import('app');
`
```

- Now import the required cultures from the installed location to app.vue file as given in the following code example.

```ts
//import the loadCldr from ej2-base
```

```
import { loadCldr} from '@syncfusion/ej2-base';
loadCldr(
require('cldr-data/supplemental/numberingSystems.json'),
require('cldr-data/main/fr-CH/ca-gregorian.json'),
require('cldr-data/main/fr-CH/numbers.json'),
require('cldr-data/main/fr-CH/timeZoneNames.json')
);
`
```

- Set the culture to Scheduler by using the locale property.

APP.VUE

```
<template>
    <div id='app'>
     <div id='container'>
      <ejs-schedule :locale = 'locale' height='550px'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
           <e-views>
              <e-view option='Day'></e-view>
              <e-view option='Week'></e-view>
              <e-view option='WorkWeek'></e-view>
              <e-view option='Month'></e-view>
            </e-views>
         </ejs-schedule>
        </div>
      </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { loadCldr } from '@syncfusion/ej2-base';
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
    import * as numberingSystems from './numberingSystems.json';
    import * as gregorian from './ca-gregorian.json';
    import * as numbers from './numbers.json';
    import * as timeZoneNames from './timeZoneNames.json';
    Vue.use(SchedulePlugin);
    loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
    export default {
        data () {
            return {
               height: '550px',
               locale: 'fr-CH',
               selectedDate: new Date(2018, 0, 15),
               eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/localization-cs1" %}

Localizing the static Scheduler text

Localization library allows to display all the static text, date content, and time mode of the Scheduler following the localized language. To achieve this, set the locale property of Scheduler, as well as define the translation text of static words of Scheduler through the load method.

For example, the following code example lets you to define the French translation words for all the static words used in Scheduler.

APP.VUE

```
<template>
    <div id='app'>
     <div id='container'>
      <ejs-schedule height='550px' :selectedDate='selectedDate'
:eventSettings='eventSettings'>
           <e-views>
              <e-view option='Day'></e-view>
              <e-view option='Week'></e-view>
              <e-view option='WorkWeek'></e-view>
              <e-view option='Month'></e-view>
            </e-views>
         </ejs-schedule>
        </div>
      </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { L10n, loadCldr, setCulture } from '@syncfusion/ej2-base';
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
    import * as localeText from './locale.json';
    import * as numberingSystems from './numberingSystems.json';
```

```
    import * as gregorian from './ca-gregorian.json';
    import * as numbers from './numbers.json';
    import * as timeZoneNames from './timeZoneNames.json';
    Vue.use(SchedulePlugin);
    L10n.load(localeText);
    loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
    setCulture('fr-CH');
    export default {
        data () {
            return {
               height: '550px',
               selectedDate: new Date(2018, 0, 15),
               eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/localization-cs2" %}

The localized words for static text used in Scheduler and Recurrence Editor can be referred from the following code.

`ts

L10n.load({

"en": {

"schedule": {

"day": "Day",

"week": "Week",

"workWeek": "Work Week",

"month": "Month",

"agenda": "Agenda",
"weekAgenda": "Week Agenda",
"workWeekAgenda": "Work Week Agenda",
"monthAgenda": "Month Agenda",
"today": "Today",
"noEvents": "No events",
"emptyContainer": "There are no events scheduled on this day.",
"allDay": "All day",
"start": "Start",
"end": "End",
"more": "more",
"close": "Close",
"cancel": "Cancel",
"noTitle": "(No Title)",
"delete": "Delete",
"deleteEvent": "Delete Event",
"deleteMultipleEvent": "Delete Multiple Events",
"selectedItems": "Items selected",
"deleteSeries": "Delete Series",
"edit": "Edit",
"editSeries": "Edit Series",
"editEvent": "Edit Event",
"createEvent": "Create",
"subject": "Subject",
"addTitle": "Add title",
"moreDetails": "More Details",
"save": "Save",
"editContent": "Do you want to edit only this event or entire series?",
"deleteRecurrenceContent": "Do you want to delete only this event or entire series?",
"deleteContent": "Are you sure you want to delete this event?",
"deleteMultipleContent": "Are you sure you want to delete the selected events?",
"newEvent": "New Event",
"title": "Title",

"location": "Location",

"description": "Description",

"timezone": "Timezone",

"startTimezone": "Start Timezone",

"endTimezone": "End Timezone",

"repeat": "Repeat",

"saveButton": "Save",

"cancelButton": "Cancel",

"deleteButton": "Delete",

"recurrence": "Recurrence",

"wrongPattern": "The recurrence pattern is not valid.",

"seriesChangeAlert": "The changes made to specific instances of this series will be cancelled and those events will match the series again.",

"createError": "The duration of the event must be shorter than how frequently it occurs. Shorten the duration, or change the recurrence pattern in the recurrence event editor.",

"recurrenceDateValidation": "Some months have fewer than the selected date. For these months, the occurrence will fall on the last date of the month.",

"sameDayAlert": "Two occurrences of the same event cannot occur on the same day.",

"editRecurrence": "Edit Recurrence",

"repeats": "Repeats",

"alert": "Alert",

"startEndError": "The selected end date occurs before the start date.",

"invalidDateError": "The entered date value is invalid.",

"ok": "Ok",

"occurrence": "Occurrence",

"series": "Series",

"previous": "Previous",

"next": "Next",

"timelineDay": "Timeline Day",

"timelineWeek": "Timeline Week",

"timelineWorkWeek": "Timeline Work Week",

"timelineMonth": "Timeline Month",

"expandAllDaySection": "Expand",

"collapseAllDaySection": "Collapse"

Copyright © 2001 -2024 Syncfusion Inc.

},
"recurrenceeditor": {
"none": "None",
"daily": "Daily",
"weekly": "Weekly",
"monthly": "Monthly",
"month": "Month",
"yearly": "Yearly",
"never": "Never",
"until": "Until",
"count": "Count",
"first": "First",
"second": "Second",
"third": "Third",
"fourth": "Fourth",
"last": "Last",
"repeat": "Repeat",
"repeatEvery": "Repeat Every",
"on": "Repeat On",
"end": "End",
"onDay": "Day",
"days": "Day(s)",
"weeks": "Week(s)",
"months": "Month(s)",
"years": "Year(s)",
"every": "every",
"summaryTimes": "time(s)",
"summaryOn": "on",
"summaryUntil": "until",
"summaryRepeat": "Repeats",
"summaryDay": "day(s)",
"summaryWeek": "week(s)",
"summaryMonth": "month(s)",

"summaryYear": "year(s)"

}

}

});

`

Setting date format

Scheduler can be used with all valid date formats and by default it follows the universal date format "MM/dd/yyyy". If the dateFormat property is not specified particularly, then it will work based on the locale that is assigned to the Scheduler. As the default locale applied on Scheduler is "en-US", this makes it to follow the "MM/dd/yyyy" pattern.

APP.VUE

```
<template>
    <div id='app'>
     <div id='container'>
      <ejs-schedule :dateFormat = 'dateFormat' height='550px'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
           <e-views>
              <e-view option='Day'></e-view>
              <e-view option='Week'></e-view>
              <e-view option='WorkWeek'></e-view>
              <e-view option='Month'></e-view>
            </e-views>
         </ejs-schedule>
        </div>
      </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
               height: '550px',
               dateFormat: 'yyyy/MM/dd',
               selectedDate: new Date(2018, 0, 15),
               eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/localization-cs3" %}

Setting the time format

Time formats is a way of representing the time value in different string formats in the Scheduler. By default, the time mode of the Scheduler can be either 12 or 24 hours format which is completely based on the `locale` set to the Scheduler. Since the default `locale` value of the Scheduler is en-US, the time mode will be set to 12 hours format automatically. You can also customize the format by using the `timeFormat` property. To know more about the time format standards, refer to the Date and Time Format section.

The following example demonstrates the Scheduler component in 24 hours format.

APP.VUE

```
<template>
    <div id='app'>
     <div id='container'>
      <ejs-schedule :timeFormat='timeFormat' height='550px'
:selectedDate='selectedDate' :eventSettings='eventSettings'>
           <e-views>
              <e-view option='Day'></e-view>
              <e-view option='Week'></e-view>
              <e-view option='WorkWeek'></e-view>
              <e-view option='Month'></e-view>
            </e-views>
         </ejs-schedule>
        </div>
      </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
               height: '550px',
               timeFormat: 'HH:mm',
               selectedDate: new Date(2018, 0, 15),
               eventSettings: { dataSource: scheduleData }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/localization-cs4" %}

Note: timeFormat property only accepts the valid time format's.

Displaying Scheduler in RTL mode

The Scheduler layout and its behavior can be changed as per the common RTL (Right to Left) conventions by setting enableRtl to true. By doing so, the Scheduler will display its usual layout from right to left. It's default value is false.

APP.VUE

```
<template>
    <div id='app'>
     <div id='container'>
      <ejs-schedule width='100%' height='550px' :selectedDate='selectedDate'
:eventSettings='eventSettings' :enableRtl='enableRtl'>
          <e-views>
             <e-view option='Day'></e-view>
             <e-view option='Week'></e-view>
             <e-view option='WorkWeek'></e-view>
           </e-views>
        </ejs-schedule>
       </div>
      </div>
</template>
<script>
    import Vue from 'vue';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, WorkWeek } from '@syncfusion/ej2-
vue-schedule';
    Vue.use(SchedulePlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    export default {
        data () {
            return {
               enableRtl: true,
               selectedDate: new Date(2018, 0, 15),
               eventSettings: { dataSource: scheduleData }
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek]
        }
    }
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/localization-cs5" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

See Also

- How to change first day of the week in the Scheduler

Accessibility in Vue Schedule component

The Scheduler has been designed based on the WAI-ARIA specifications, thus applying the appropriate ARIA roles, states and properties for the Scheduler elements. It is also available with a built-in keyboard navigation support, making it easier for the people who use assistive technologies or who completely rely on the Keyboard support. As per the accessibility standard, the navigated dates, views and other interactive actions performed on the Scheduler will be read out to the target users who use assistive technologies such as screen readers.

The Scheduler makes use of the most required ARIA attributes such as `aria-label` and `role` to support the accessibility in it. To be more accurate, it must be used with an ARIA compliant browser along with the screen reader running from backend.

The accessibility compliance for the Schedule component is outlined below.

Copyright © 2001 -2024 Syncfusion Inc.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

ARIA attributes

The Scheduler parent element is assigned with a role of main, to denote it as the main content of a component as well as a unique element of the entire document.

The following ARIA attributes are used in the Scheduler.

| Attributes | Description |

|-------|---------|

| role="main" | Attribute added to the Scheduler element describes the actual role of the element and denote it as a main and unique content. |

| role="button" | Attribute is assigned to the appointments of Scheduler, to denote it as a clickable element. |

| aria-label | Attribute is set to the Scheduler parent element and its default value is Scheduler's current date. On every time, the date is navigated, this attribute is updated with appropriate current date values. It is also assigned to other scheduler UI elements such as previous and next date navigation buttons depicting its purpose, div element displaying date range in the header bar and appointment elements. |

| aria-labelledby | It indicates editor dialog title to the user through assistive technologies. |

| aria-describedby | It indicates editor dialog content description to the user through assistive technologies. |

| aria-disabled | Attribute is set to the appointment element to indicates the disabled state of the Scheduler.

Keyboard interaction

All the Scheduler actions can be controlled via keyboard keys by using the `allowKeyboardInteraction` property which is set to `true` by default. The following are the standard keys that work on Scheduler.

| Keys | Description |

|-----|-----|

| Alt + j | Focuses the Scheduler element [provided from application end]. |

| Tab | Focuses the first or active item on the Scheduler header bar and then move the focus to the next available event elements. If no events present, then focus moves out of the component. |

| Shift + Tab | Reverse focusing of the `Tab` key functionality. Inverse focusing of event elements from the last one and then move onto the first or active item on Scheduler header bar and then moves out of the component.

| Enter | Opens the quick info popup on the selected cells or events. |

| Escape | Closes any of the popup that are in open state. |

| Arrow | To move onto the next available cells in either of the needed directions. (left, right, top and right) |

| Shift + Arrow | For multiple cell selection on either direction. |

| Delete | Deletes one or more selected events. |

| Ctrl + Click on events | To select multiple events. |

| Alt + Number (from 1 to 6) |To switch between the views of Scheduler. |

| Ctrl + Left Arrow | To navigate to the previous date period. |

| Ctrl + Right Arrow | To navigate to the next date period. |

Copyright © 2001 -2024 Syncfusion Inc.

| Left or Right Arrow | On pressing any of these keys, when focus is currently on the Schedule header bar, moves the focus to the previous or next items in the header bar. |

| Space or Enter | It activates any of the focused items. |

| Page Up & Page Down | To scroll through the work cells area. |

| Home | To move the selection to the first cell of Scheduler. |

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Ensuring accessibility

The Scheduler component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Scheduler component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Scheduler component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/schedule.html" %}

See also

- Accessibility in Syncfusion Vue components

Scheduler styling in Vue Schedule component

To modify the Scheduler appearance, you need to override the default CSS of Scheduler. Also, you have an option to create your own custom theme using our Theme Studio. Please find the list of CSS classes in Scheduler.

| Css class | Purpose |

|-------|---------|

| .e-schedule .e-vertical-view .e-work-cells | Work cells in vertical views of scheduler |

| .e-schedule .e-month-view .e-work-cells | Work cells in month view of scheduler |

| .e-schedule .e-month-view .e-other-month | Work cells of other month in month view of scheduler |

| .e-schedule .e-timeline-view .e-work-cells | Work cells in timeline views of scheduler |

| .e-schedule .e-timeline-month-view .e-work-cells | Work cells in timeline month view of scheduler |

| .e-schedule .e-timeline-year-view .e-work-cells | Work cells in timeline year view of scheduler |

| .e-schedule .e-timeline-year-view .e-work-cells.e-other-month | Work cells of other month in timeline year view of scheduler |

| .e-schedule .e-month-agenda-view .e-work-cells | Work cells in month agenda view of scheduler |

| .e-schedule .e-month-agenda-view .e-other-month | Work cells of other month in month agenda view of scheduler |

| .e-schedule .e-year-view .e-calendar-wrapper .e-month-calendar.e-calendar .e-other-month | Work cells of other month in year view of scheduler |

Copyright © 2001 -2024 Syncfusion Inc.

| .e-schedule .e-vertical-view .e-all-day-cells | All day cells in vertical views of scheduler |

| .e-schedule .e-vertical-view .e-work-hours | Work hour cells in vertical views of scheduler |

| .e-schedule .e-month-view .e-work-days | Work day cells in month view of scheduler |

| .e-schedule .e-month-agenda-view .e-work-days | Work day cells in month agenda view of scheduler |

| .e-schedule .e-timeline-view .e-work-hours | Work hour cells in timeline views of scheduler |

| .e-schedule .e-timeline-month-view .e-work-days | Work day cells in timeline month view of scheduler |

| .e-schedule .e-timeline-year-view .e-work-cells.e-work-days | Work day cells in timeline year view of scheduler |

| .e-schedule .e-vertical-view .e-day-wrapper .e-appointment | Appointment in vertical views of scheduler |

| .e-schedule .e-vertical-view .e-all-day-appointment-wrapper .e-appointment | All day Appointment in vertical views of scheduler |

| .e-schedule .e-month-view .e-appointment | Appointment in month view of scheduler |

| .e-schedule .e-timeline-view .e-appointment | Appointment in timeline views of scheduler |

| .e-schedule .e-timeline-month-view .e-appointment | Appointment in timeline month view of scheduler |

| .e-schedule .e-timeline-year-view .e-event-table .e-appointment | Appointment in timeline year view of scheduler |

| .e-schedule .e-year-view .e-calendar-wrapper .e-month-calendar.e-calendar .e-appointment | Appointment in year view of scheduler |

| .e-schedule .e-agenda-view .e-appointment | Appointment in agenda view of scheduler |

| .e-schedule .e-month-agenda-view .e-appointment-indicator | Appointment in month agenda view of scheduler |

| .e-schedule .e-block-appointment | Block appointment in scheduler |

| .e-schedule .e-read-only | Read only appointment in scheduler. |

| e-appointment-border | Appointment which are currently selected, use the appointment class hierarchical based on your views. |

| e-selected-cells | work cells which are currently selected, use the work cell class hierarchical based on your views. |

| e-header-cells | Header cells of scheduler, use the work cells hierarchical based on your views. |

| .e-schedule .e-vertical-view .e-resource-cells| Resource cells in vertical views of scheduler. |

| .e-schedule .e-month-view .e-resource-cells| Resource cells in month view of scheduler. |

| .e-schedule .e-timeline-view .e-resource-cells | Resource cells in timeline views of scheduler. |

| .e-schedule .e-timeline-month-view .e-resource-cells| Resource cells in timeline month view of scheduler. |

| e-parent-node | Parent resource cells in timeline views of scheduler. |

Copyright © 2001 -2024 Syncfusion Inc.

| e-child-node | Child resource cells in timeline views of scheduler. |

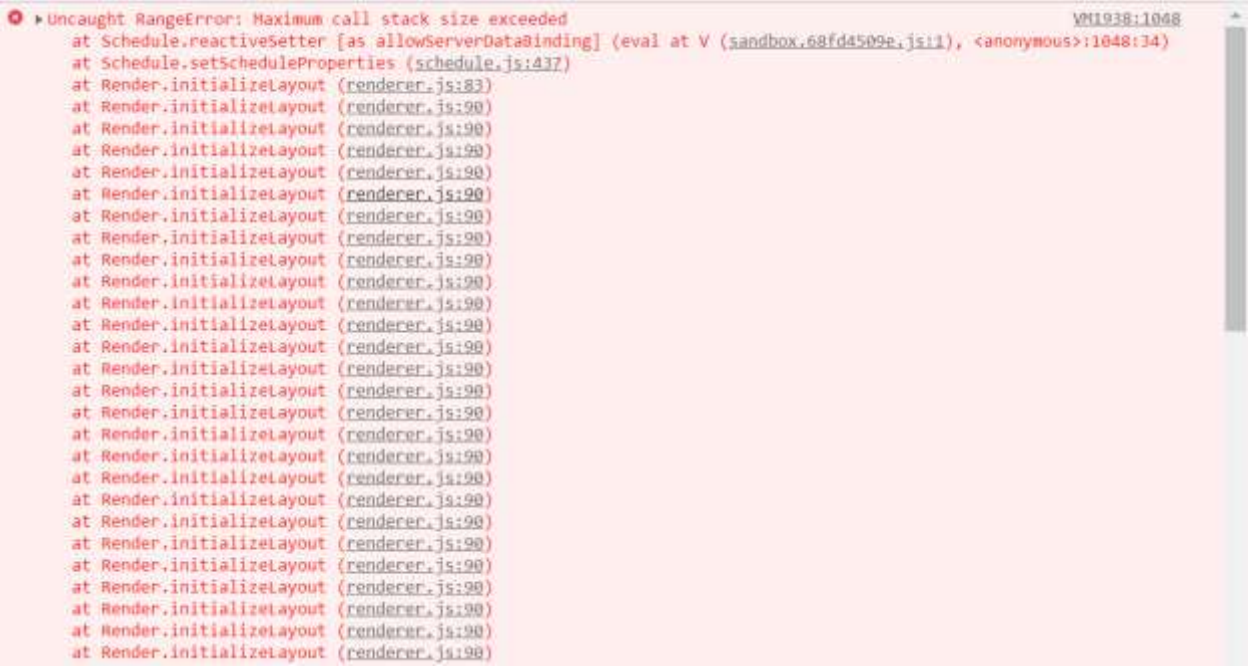
You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Frequently asked questions in Vue Schedule component

In this article, you can find some frequently asked questions and corresponding solutions while getting hands-on experience with scheduler control.

Maximum call stack size exceeded

Error Image:



Solution:

The above error occurs when using scheduler views that were not imported into the project. You can resolve this issue by importing the required view modules.

In the below code, Day option is used without injecting, So, it throws the above error. You can resolve this problem by simply injecting the day module in below code.

`

<template>

<ejs-schedule id="Schedule" width="100%" height="550px">

<e-views>

<e-view option="Day"></e-view>

<e-view option="TimelineWeek"></e-view>

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-view option="TimelineMonth"></e-view>
<e-view option="Agenda"></e-view>
</e-views>
</ejs-schedule>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import {
SchedulePlugin,
TimelineViews,
TimelineMonth,
Agenda,
} from "@syncfusion/ej2-vue-schedule";
`
```

Grouping with empty resources

Grouping without providing any resource data will throw the following problems.

- Normal(vertical) views are rendered, but you are not able to perform CRUD operations
- Timeline views not at all render and shows empty scheduler table

So, we suggest to avoid grouping with empty resources in the scheduler.

Not providing e-field in editor template

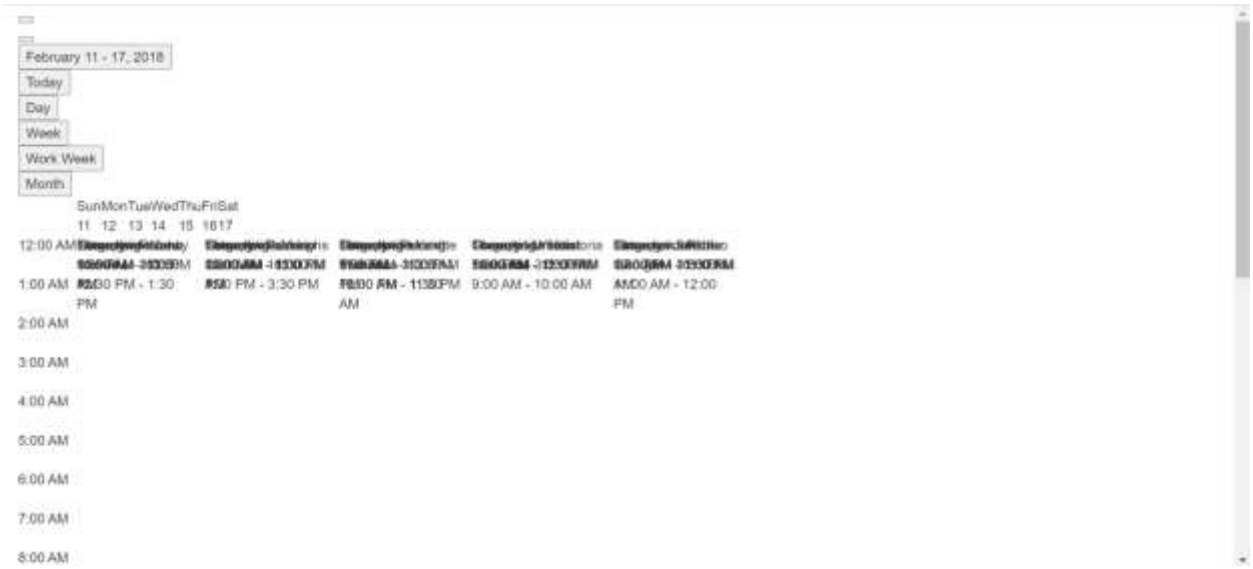
Error: While using editor template, value of `e-field` is missing in editor window.

Solution: `e-field` value is mandatory, we need to add it. Please refer here for more info.

Missing CSS reference

Error Image:

Copyright © 2001 -2024 Syncfusion Inc.



Solution:

The above problem occurs when missing CSS references for the scheduler in a project. You can resolve this issue by providing proper CSS for the scheduler.

`

<! — scheduler CSS is referred from this link —>

<link href="https://cdn.syncfusion.com/ej2/material.css" rel="stylesheet">

or

<! — From here —>

<style>

@import "/node_modules/@syncfusion/ej2-base/styles/material.css";

@import "/node_modules/@syncfusion/ej2-buttons/styles/material.css";

@import "/node_modules/@syncfusion/ej2-calendars/styles/material.css";

@import "/node_modules/@syncfusion/ej2-dropdowns/styles/material.css";

@import "/node_modules/@syncfusion/ej2-inputs/styles/material.css";

@import "/node_modules/@syncfusion/ej2-navigations/styles/material.css";

@import "/node_modules/@syncfusion/ej2-popups/styles/material.css";

@import "/node_modules/@syncfusion/ej2-vue-schedule/styles/material.css";

</style>

`

QuickInfoTemplate at bottom

When using the quickInfoTemplate in scheduler, sometimes quickinfo popup not shown fully at the bottom area of scheduler. You can resolve this by using cellClick and eventClick events and below code snippet.

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<ejs-schedule
id="Schedule"
width="100%"
height="550px"
:cellClick="onClick"
:eventClick="onClick"

</ejs-schedule>
</template>
.
.
var eventAdded = false;
methods: {
onClick: function () {
if (!this.eventAdded) {
let popupInstance = document.querySelector('.e-quick-popup-wrapper').ej2_instances[0];
popupInstance.open = () => {
popupInstance.refreshPosition();
};
this.eventAdded = true;
}
}
```

Not importing culture files while using localization

Error Image:

```
ERROR TypeError: Cannot read property 'trim' of undefined                errors.ts:30
    at t.e.formatDateRange (util.js:6)
    at t.e.getDateRangeText (util.js:6)
    at e.updateHeader (util.js:6)
    at e.initializeLayout (util.js:6)
    at e.render (util.js:6)
    at ScheduleComponent.n.initializeView (util.js:6)
    at ScheduleComponent.n.renderElements (util.js:6)
    at ScheduleComponent.n.initializeResources (util.js:6)
    at ScheduleComponent.n.render (util.js:6)
    at ScheduleComponent.r.appendTo (dom.js:461)
```

Copyright © 2001 -2024 Syncfusion Inc.

While using locale in scheduler, not importing the required culture files properly throws the problem.

Solution: Properly add and import the culture files(numberingSystems, timeZoneNames, loadCldr, L10n etc.,) in your project will resolve the problem.

`javascript

import { loadCldr, L10n } from '@syncfusion/ej2-base';

import * as numberingSystems from './culture-files/numberingSystems.json';

import * as gregorian from './culture-files/ca-gregorian.json';

import * as numbers from './culture-files/numbers.json';

import * as timeZoneNames from './culture-files/timeZoneNames.json';

loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);

L10n.load({

'en-GB': {

schedule: {

day: 'Day',

week: 'Week',

workWeek: 'Work Week',

month: 'Month'

}

}

});

`

How To

Add edit and remove events in Vue Schedule component

CRUD actions can be manually performed on appointments using `addEvent`, `saveEvent` and `deleteEvent` methods as shown below.

Normal event

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <tr>
                    <td>
                        <div>
                            <ejs-button v-on:click.native='onAdd'>Add</ejs-
button>
                        </div>
                    </td>
                    <td>
```

```
                        <div>
                            <ejs-button v-
on:click.native='onSave'>Edit</ejs-button>
                        </div>
                    </td>
                    <td>
                        <div>
                            <ejs-button v-
on:click.native='onDelete'>Delete</ejs-button>
                        </div>
                    </td>
                </tr><br>
                <ejs-schedule id='Schedule' width='100%' height='485px'
:eventSettings='eventSettings' :selectedDate='selectedDate'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                        <e-view option='Month'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { extend } from '@syncfusion/ej2-base';
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
    import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
    Vue.use(ButtonPlugin);
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: {
                    dataSource:  [{
                            Id: 3,
                            Subject: 'Testing',
                            StartTime: new Date(2018, 1, 11, 9, 0),
                            EndTime: new Date(2018, 1, 11, 10, 0),
                            IsAllDay: false
                        },{
                            Id: 4,
                            Subject: 'Vacation',
                            StartTime: new Date(2018, 1, 13, 9, 0),
                            EndTime: new Date(2018, 1, 13, 10, 0),
                            IsAllDay: false
                        }]
                },
                selectedDate: new Date(2018, 1, 15)
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        methods: {
            onSave: function () {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                let eventData = {
                    Id: 3,
                    Subject: 'Testing-edited',
                    StartTime: new Date(2018, 1, 11, 10, 0),
                    EndTime: new Date(2018, 1, 11, 11, 0),
                    IsAllDay: false
                };
                scheduleObj.saveEvent(eventData);
            },
            onAdd: function () {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                let Data = [{
                    Id: 1,
                    Subject: 'Conference',
                    StartTime: new Date(2018, 1, 12, 9, 0),
                    EndTime: new Date(2018, 1, 12, 10, 0),
                    IsAllDay: false
                },{
                    Id: 2,
                    Subject: 'Meeting',
                    StartTime: new Date(2018, 1, 15, 10, 0),
                    EndTime: new Date(2018, 1, 15, 11, 30),
                    IsAllDay: false
                }];
                scheduleObj.addEvent(Data);
            },
            onDelete: function () {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                scheduleObj.deleteEvent(4);
            }
        }
    }
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/app-crud-cs1" %}

Recurrence event

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <tr>
                    <td>
                        <div>
                            <ejs-button v-on:click.native='onAdd'>Add</ejs-
button>
                        </div>
                    </td>
                    <td>
                        <div>
                            <ejs-button v-
on:click.native='onSave'>Edit</ejs-button>
                        </div>
                    </td>
                    <td>
                        <div>
                            <ejs-button v-
on:click.native='onDelete'>Delete</ejs-button>
                        </div>
                    </td>
                </tr><br>
                <ejs-schedule id='Schedule' width='100%' height='485px'
:eventSettings='eventSettings' :selectedDate='selectedDate'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                        <e-view option='Month'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { extend } from '@syncfusion/ej2-base';
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
    import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
    Vue.use(ButtonPlugin);
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: {
                    dataSource: [{
```

```
                        Id: 3,
                        Subject: 'Testing',
                        StartTime: new Date(2018, 1, 11, 9, 0),
                        EndTime: new Date(2018, 1, 11, 10, 0),
                        IsAllDay: false,
                        RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=3',
                    },{
                        Id: 4,
                        Subject: 'Vacation',
                        StartTime: new Date(2018, 1, 12, 11, 0),
                        EndTime: new Date(2018, 1, 12, 12, 0),
                        IsAllDay: false,
                        RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2',
                    }]
                },
                selectedDate: new Date(2018, 1, 15),
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        },
        methods: {
            onSave: function () {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                let eventData = {
                    Id: 3,
                    Subject: 'Testing-edited',
                    StartTime: new Date(2018, 1, 11, 10, 0),
                    EndTime: new Date(2018, 1, 11, 11, 0),
                    IsAllDay: false,
                    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=3',
                };
                scheduleObj.saveEvent(eventData,'EditOccurrence');
            },
            onAdd: function () {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                let Data = [{
                    Id: 1,
                    Subject: 'Conference',
                    StartTime: new Date(2018, 1, 15, 9, 0),
                    EndTime: new Date(2018, 1, 15, 10, 0),
                    IsAllDay: false,
                    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2',
                }];
                scheduleObj.addEvent(Data);
            },
            onDelete: function () {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                let scheduleData = [{
                    Id: 4,
                    Subject: 'Vacation',
                    RecurrenceID: 4,
                    StartTime: new Date(2018, 1, 12, 11, 0),
                    EndTime: new Date(2018, 1, 12, 12, 0),
```

```
                    IsAllDay: false,
                    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2',
                }];
                scheduleObj.deleteEvent(scheduleData,'DeleteSeries');
            },
        }
    }
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/app-crud-cs2" %}

Set default value for event fields in Vue Schedule component

Event window default fields name like Title, Location, etc.. can be customized and default value can be set to Subject field using default property which will be added if an appointment is created with empty subject.

APP.VUE

```
<template>
    <div>
        <div class='col-md-9 control-section'>
            <div class='content-wrapper'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { extend } from '@syncfusion/ej2-base';
```

```
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: {
                    dataSource: extend([], scheduleData, null, true),
                    fields: {
                        subject: { title: 'Event Name', name: 'Subject',
default: 'Add Name' },
                        location: { title: 'Event Location', name:
'Location', default: 'USA'},
                        description: { title: 'Summary', name: 'Description'
},
                        startTime: { title: 'From', name: 'StartTime' },
                        endTime: { title: 'To', name: 'EndTime' }
                    }
                },
                selectedDate: new Date(2018, 1, 15)
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda]
        }
    }
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/how-to-cs2" %}

Open event editor manually in Vue Schedule component

Open Editor Window externally

Scheduler allows the user to manually open the event editor on specific time or on certain events using openEditor method. To open the editor on specific range of time, user need to pass the cell details as first argument and **Add** as second argument whereas to open it on event pass that event detail and **Save** as arguments. In the following code example, on clicking the respective button will open the respective editor window manually.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <tr>
                    <td>
                        <div>
                            <ejs-button  v-on:click.native='onSubmit'>Click
to open Editor</ejs-button>
                        </div>
                    </td>
                    <td>
                        <div>
                            <ejs-button v-
on:click.native='onEventSubmit'>Click to open Event Editor</ejs-button>
                        </div>
                    </td>
                </tr>
                <br><br>
                <ejs-schedule id='Schedule' width='100%' height='500px'
:eventSettings='eventSettings' :selectedDate='selectedDate'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                        <e-view option='Month'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { extend } from '@syncfusion/ej2-base';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
    import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
    Vue.use(ButtonPlugin);
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: { dataSource: extend([], scheduleData, null,
true)  },
                selectedDate: new Date(2018, 1, 15)
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        },
        methods: {
            onEventSubmit: function () {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                let eventData ={
                    Id: 4,
                    Subject: 'Meteor Showers in 2018',
                    StartTime: new Date(2018, 1, 14, 13, 0),
                    EndTime: new Date(2018, 1, 14, 14, 30)
                };
                scheduleObj.openEditor(eventData,'Save');
            },
            onSubmit: function () {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                let cellData ={
                      startTime: new Date(2018, 1, 15, 10, 0),
                      endTime: new Date(2018, 1, 15, 11, 0),
                };
                scheduleObj.openEditor(cellData,'Add');
            }
        }
    }
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/open-editor-cs1" %}

Open editor window on single click

By default, Scheduler Editor window will open when double clicking the cells or appointments. You can also open the editor window with single click by using openEditor method in eventClick and cellClick events of scheduler and setting false to showQuickInfo. The following example shows how to open editor window on single click of cells and appointments.

APP.VUE

```
<template>
    <div id="app">
        <div id="container">
            <ejs-schedule ref='scheduleObj' width="100%" height="550px"
:eventSettings='eventSettings' :selectedDate='selectedDate'
            :cellClick='onCellClick'
```

```
            :eventClick='onEventClick'
            :showQuickInfo='showQuickInfo'
            >
                <e-views>
                    <e-view option="Day"></e-view>
                    <e-view option="Week"></e-view>
                    <e-view option="WorkWeek"></e-view>
                    <e-view option="Month"></e-view>
                    <e-view option="Agenda"></e-view>
                </e-views>
            </ejs-schedule>
        </div>
    </div>
</template>
<script>
    import Vue from "vue";
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
"@syncfusion/ej2-vue-schedule";
    import { schedulerData } from './datasource.js';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: {
                    dataSource: schedulerData
                },
                selectedDate: new Date(2021, 7, 15),
                showQuickInfo: false,
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda]
        },
        methods: {
            onCellClick: function(args) {
                this.$refs.scheduleObj.openEditor(args, 'Add');
            },
            onEventClick: function(args) {
               if (!(args.event as any).RecurrenceRule) {
                this.$refs.scheduleObj.openEditor(args.event, 'Save');
                }
                else {
                this.$refs.scheduleObj.quickPopup.openRecurrenceAlert();
                }
            }
        }
    }
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-
navigations/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-popups/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/default-cs4" %}

Prevent date navigation in Vue Schedule component

We can prevent navigation while clicking on the date header by simply removing e-navigate class from header cells which can be achieved in the renderCell event as shown in the following code example.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <ejs-schedule id='Schedule' width='100%' height='550px'
:eventSettings='eventSettings' :selectedDate='selectedDate'
:renderCell='onRenderCell'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                        <e-view option='Month'></e-view>
                        <e-view option='Agenda'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { extend, removeClass } from '@syncfusion/ej2-base';
    import { scheduleData } from './datasource.js';
    import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: { dataSource: extend([], scheduleData, null,
true)  },
                selectedDate: new Date(2018, 1, 15)
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month, Agenda]
        },
        methods: {
            onRenderCell: function (args) {
                if(args.elementType === 'dateHeader' || args.elementType ===
'monthCells') {
                    removeClass(args.element.childNodes, 'e-navigate');
                }
            }
```

```
        }
    }
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/how-to-cs1" %}

Half yearly view in Vue Schedule component

The year view of our scheduler displays all the 365 days and their related appointments of a particular year. You can customize the year view by using the following properties.

- firstMonthOfYear
- monthsCount
- monthHeaderTemplate

In the following code example, you can see how to render only the last six months of a year in the scheduler. To start with the month of June, `firstMonthYear` is set to 6 and `monthsCount` is set to 6 to render only 6 months.

APP.VUE

```
<template>
  <div>
    <div id="app">
      <div id="container">
        <ejs-schedule
          id="Schedule"
          height="650px"
          :eventSettings="eventSettings"
          :currentView="currentView"
          :firstMonthOfYear="firstMonthOfYear"
          :monthsCount="monthsCount"
          :group="group"
          :monthHeaderTemplate="monthHeaderTemplate"
          :resourceHeaderTemplate="resourceHeaderTemplate"
        >
          <e-views>
            <e-view option="Year"></e-view>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <e-view
              option="TimelineYear"
              displayName="Horizontal TimelineYear"
              isSelected="true"
            ></e-view>
            <e-view
              option="TimelineYear"
              displayName="Vertical TimelineYear"
              orientation="Vertical"
            ></e-view>
          </e-views>
          <e-resources>
            <e-resource
              field="OwnerId"
              title="Owner"
              name="Owners"
              :dataSource="ownerDataSource"
              textField="OwnerText"
              idField="Id"
              colorField="OwnerColor"
            ></e-resource>
          </e-resources>
        </ejs-schedule>
      </div>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { resourceData } from "./datasource.js";
import {
  SchedulePlugin,
  Year,
  TimelineYear,
  Resize,
  DragAndDrop
} from "@syncfusion/ej2-vue-schedule";
Vue.use(SchedulePlugin);
var monthHeaderTemplate = Vue.component("demo", {
  template: "<div>{{monthHeaderTemplate(data.date)}}</div>",
  data() {
    return {
      data: {},
      monthHeaderTemplate: function(date) {
        return (
          date.toLocaleString("en-us", { month: "long" }) +
          " " +
          date.getFullYear()
        );
      }
    };
  }
});
var resourceHeaderTemplateVue = Vue.component("headerTemplate", {
  template: `<div class='template-wrap'><div class="resource-details"><div
class="resource-name">{{data.resourceData.OwnerText}}</div></div></div>`,
  data() {
```

```
    return {
      data: {}
    };
  }
});
export default {
  data() {
    return {
      width: "100%",
      height: "550px",
      currentView: "TimelineWeek",
      selectedDate: new Date(2021, 7, 4),
      firstMonthOfYear: 6,
      monthsCount: 6,
      group: {
        resources: ["Owners"]
      },
      monthHeaderTemplate: function(e) {
        return { template: monthHeaderTemplate };
      },
      allowMultiple: true,
      ownerDataSource: [
        { OwnerText: "Nancy", Id: 1, OwnerColor: "#ffaa00" },
        { OwnerText: "Steven", Id: 2, OwnerColor: "#f8a398" },
        { OwnerText: "Robert", Id: 3, OwnerColor: "#7499e1" },
        { OwnerText: "Smith", Id: 4, OwnerColor: "#5978ee" },
        { OwnerText: "Micheal", Id: 5, OwnerColor: "#df5286" }
      ],
      resourceHeaderTemplate: function(e) {
        return {
          template: resourceHeaderTemplateVue
        };
      },
      eventSettings: { dataSource: resourceData }
    };
  },
  provide: {
    schedule: [Year, TimelineYear, Resize, DragAndDrop]
  }
};
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
.e-schedule .e-vertical-view .e-resource-cells {
  height: 62px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
.e-schedule .template-wrap {
  display: flex;
  text-align: left;
}
.e-schedule .template-wrap .resource-details {
  padding-left: 10px;
}
.e-schedule .template-wrap .resource-details .resource-name {
  font-size: 16px;
  font-weight: 500;
  margin-top: 5px;
}
.e-schedule.e-device .template-wrap .resource-details .resource-name {
  font-size: inherit;
  font-weight: inherit;
}
.e-schedule.e-device .e-resource-tree-popup .e-fullrow {
  height: 50px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/year-customizations-cs1" %}

Set different work hours in Vue Schedule component

By default, the work hours of the Scheduler is highlighted based on the start and end values provided within the workHours property which remains same for all days. To highlight different work hours range for different days,setWorkHours method. You can pass date object/ multiple date objects collection as first argument and start and end time need to be added as work hours should be passed as second and third arguments respectively. In the following code example, on button click 11:00 AM to 08:00 PM of 15th and 17th February has been added in work hours.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <tr>
                    <td>
                        <div>
                            <ejs-button id='btn1' v-
on:click.native='onChangeWorkHours'>Change the Work Hours</ejs-button>
                        </div>
                    </td>
                </tr>
                <ejs-schedule ref='scheduleObj' width='100%' height='500px'
:eventSettings='eventSettings' :selectedDate='selectedDate'
:workHours='workHours'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                        <e-view option='Month'></e-view>
                    </e-views>
```

```
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { extend } from '@syncfusion/ej2-base';
    import { scheduleData } from './datasource.js';
    import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
    Vue.use(ButtonPlugin);
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: { dataSource: extend([], scheduleData, null,
true)  },
                workHours: {
                    highlight: true,
                    start: '09:00',
                    end: '11:00'
                },
                selectedDate: new Date(2018, 1, 15)
            }
        },
        provide: {
            schedule: [Day, Week, WorkWeek, Month]
        },
        methods: {
            onChangeWorkHours: function () {
                let dates = [new Date(2018, 1, 15), new Date(2018, 1, 17)];
                this.$refs.scheduleObj.setWorkHours(dates, '11:00','20:00');
            }
        }
    }
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/schedule/open-editor-cs2" %}

Show quick info template in Vue Schedule component

This demo showcases the quick popups for cells and appointments with the customized templates.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
      <ejs-schedule :height='height' :width='width' :views='views'
        :selectedDate='selectedDate' :eventSettings='eventSettings'
        :quickInfoTemplates='quickInfoTemplates'>
        <e-resources>
          <e-resource field="RoomId" title="Room Type" name="MeetingRoom"
textField="Name" idField="Id" colorField="Color" :dataSource="roomData"></e-
resource>
        </e-resources>
      </ejs-schedule>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import {SchedulePlugin, Day, Week, WorkWeek, Month, Agenda  } from
'@syncfusion/ej2-vue-schedule';
import { extend, Internationalization } from "@syncfusion/ej2-base";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { TextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
import { quickInfoData } from './datasource.js';
Vue.use(SchedulePlugin);
Vue.use(ButtonPlugin);
Vue.use(DropDownListPlugin);
Vue.use(TextBoxPlugin);
var resourceData = [
  { Name: "Jammy", Id: 1, Capacity: 20, Color: "#ea7a57", Type: "Conference"
},
  { Name: "Tweety", Id: 2, Capacity: 7, Color: "#7fa900", Type: "Cabin" },
  { Name: "Nestle", Id: 3, Capacity: 5, Color: "#5978ee", Type: "Cabin" },
  { Name: "Phoenix", Id: 4, Capacity: 15, Color: "#fec200", Type:
"Conference" },
  { Name: "Mission", Id: 5, Capacity: 25, Color: "#df5286", Type:
"Conference" },
  { Name: "Hangout", Id: 6, Capacity: 10, Color: "#00bdae", Type: "Cabin" },
  { Name: "Rick Roll", Id: 7, Capacity: 20, Color: "#865fcf", Type:
"Conference" },
  { Name: "Rainbow", Id: 8, Capacity: 8, Color: "#1aaa55", Type: "Cabin" },
  { Name: "Swarm", Id: 9, Capacity: 30, Color: "#df5286", Type: "Conference"
},
  { Name: "Photogenic", Id: 10, Capacity: 25, Color: "#710193", Type:
"Conference" }
];
var headerTemplateVue = Vue.component('headerTemplate', {
  template: `<div class="quick-info-header">
    <div class="quick-info-header-content" v-if="data.elementType == 'cell'"
:style="{'align-items':'center','color':'#919191'}">
```

```
      <div class="quick-info-title">{{getHeaderTitle(data)}}</div>
      <div class="duration-text">{{getHeaderDetails(data)}}</div>
    </div>
    <div class="quick-info-header-content" v-else :style="{'background':
getHeaderStyles(data),'color':'#FFFFFF'}">
      <div class="quick-info-title">{{getHeaderTitle(data)}}</div>
      <div class="duration-text">{{getHeaderDetails(data)}}</div>
    </div>
  </div>`,
  data() {
    return {
        intl: new Internationalization(),
      data: {}
    };
  },
  methods: {
    getHeaderStyles: function(data) {
      const scheduleObj = document.querySelector(".e-
schedule").ej2_instances[0];
      const resources = scheduleObj.getResourceCollections().slice(-1)[0];
      const resourceData = resources.dataSource.filter(resource =>
resource.Id === data.RoomId)[0];
      return resourceData.Color;
    },
    getHeaderTitle: function(data) {
      return data.elementType === "cell" ? "Add Appointment" : "Appointment
Details";
    },
    getHeaderDetails: function(data) {
      return (
        this.intl.formatDate(data.StartTime, { type: "date", skeleton:
"full" }) + " (" +
        this.intl.formatDate(data.StartTime, { skeleton: 'hm' }) + " - " +
        this.intl.formatDate(data.EndTime, { skeleton: 'hm' }) + ")"
      );
    }
  }
});
var contentTemplateVue = Vue.component('contentTemplate', {
  template: `<div class="quick-info-content"><div class="e-cell-content" v-
if="data.elementType === 'cell'">
    <div class="content-area"><ejs-textbox ref="titleObj" id="title"
placeholder="Title"></ejs-textbox></div>
    <div class="content-area"><ejs-dropdownlist ref="eventTypeObj"
id="eventType" :dataSource="roomData" :index="0" :fields="fields"
    popupHeight="200px" placeholder="Choose Type"></ejs-dropdownlist></div>
    <div class="content-area"><ejs-textbox ref="notesObj" id="notes"
placeholder="Notes"></ejs-textbox></div></div>
    <div class="event-content" v-else><div class="meeting-type-
wrap"><label>Subject</label>:<span>{{data.Subject}}</span></div>
    <div class="meeting-subject-
wrap"><label>Type</label>:<span>{{getEventType(data)}}</span></div>
    <div class="notes-
wrap"><label>Notes</label>:<span>{{data.Description}}</span></div></div></di
v>`,
  data() {
    return {
```

```
        fields: { text: "Name", value: "Id" },
      roomData: extend([], resourceData, undefined, true),
      data: {}
    };
  },
  methods: {
    getEventType: function(data) {
      const scheduleObj = document.querySelector(".e-
schedule").ej2_instances[0];
      const resources = scheduleObj.getResourceCollections().slice(-1)[0];
      const resourceData = resources.dataSource.filter(resource =>
resource.Id === data.RoomId)[0];
      return resourceData.Name;
    }
  }
});
var footerTemplateVue = Vue.component('footerTemplate', {
  template: `<div class="quick-info-footer"><div class="cell-footer" v-
if="data.elementType === 'cell'">
    <ejs-button id="more-details" cssClass="e-flat" content="More Details"
v-on:click.native="buttonClickActions"></ejs-button>
    <ejs-button id="add" cssClass="e-flat" content="Add" :isPrimary="true"
v-on:click.native="buttonClickActions"></ejs-button>
    </div><div class="event-footer" v-else>
    <ejs-button id="delete" cssClass="e-flat" content="Delete" v-
on:click.native="buttonClickActions"></ejs-button>
    <ejs-button id="more-details" cssClass="e-flat" content="More Details"
:isPrimary="true" v-on:click.native="buttonClickActions"></ejs-button>
    </div></div>`,
  data() {
    return {
      data: {}
    };
  },
    methods: {
    buttonClickActions: function(e) {
      const scheduleObj = document.querySelector(".e-
schedule").ej2_instances[0];
      const quickPopup = scheduleObj.element.querySelector(".e-quick-popup-
wrapper");
      const getSlotData = function() {
        const titleObj =
quickPopup.querySelector("#title").ej2_instances[0];
        const notesObj =
quickPopup.querySelector("#notes").ej2_instances[0];
        const eventTypeObj =
quickPopup.querySelector("#eventType").ej2_instances[0];
        const cellDetails =
scheduleObj.getCellDetails(scheduleObj.getSelectedElements());
        let addObj = {};
        addObj.Id = scheduleObj.getEventMaxID();
        addObj.Subject = titleObj.value;
        addObj.StartTime = new Date(+cellDetails.startTime);
        addObj.EndTime = new Date(+cellDetails.endTime);
        addObj.Description = notesObj.value;
        addObj.RoomId = eventTypeObj.value;
        return addObj;
```

```
      };
      if (e.target.id === "add") {
        const addObj = getSlotData();
        scheduleObj.addEvent(addObj);
      } else if (e.target.id === "delete") {
        const eventDetails = scheduleObj.activeEventData.event;
        let currentAction = "Delete";
        if (eventDetails.RecurrenceRule) {
          currentAction = "DeleteOccurrence";
        }
        scheduleObj.deleteEvent(eventDetails, currentAction);
      } else {
        const isCellPopup =
quickPopup.firstElementChild.classList.contains("e-cell-popup");
        const eventDetails = isCellPopup ? getSlotData() :
scheduleObj.activeEventData.event;
        let currentAction = isCellPopup ? "Add" : "Save";
        if (eventDetails.RecurrenceRule) {
          currentAction = "EditOccurrence";
        }
        scheduleObj.openEditor(eventDetails, currentAction, true);
      }
      scheduleObj.closeQuickInfoPopup();
    }
  }
});
export default {
  data() {
    return {
      height: '650px',
      width: '100%',
      eventSettings: {
        dataSource: quickInfoData
      },
      selectedDate: new Date(2020, 0, 9),
      views: ['Day', 'Week', 'WorkWeek'],
      roomData: extend([], resourceData, undefined, true),
      quickInfoTemplates: {
        header: function(e) {
          return {
            template: headerTemplateVue
          };
        },
        content: function(e) {
          return {
            template: contentTemplateVue
          };
        },
        footer: function(e) {
          return {
            template: footerTemplateVue
          };
        }
      }
    };
  },
  provide: {
```

```
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css";
  .quick-info-header {
  background-color: white;
  padding: 8px 18px;
}
.quick-info-header-content {
  justify-content: flex-end;
  display: flex;
  flex-direction: column;
  padding: 5px 10px 5px;
}
.quick-info-title {
  font-weight: 500;
  font-size: 16px;
  letter-spacing: 0.48px;
  height: 22px;
}
.duration-text {
  font-size: 11px;
  letter-spacing: 0.33px;
  height: 14px;
}
.content-area {
  margin: 0;
  padding: 10px;
  width: auto;
}
.event-content {
  height: 90px;
  display: flex;
  flex-direction: column;
  justify-content: center;
  padding: 0 15px;
}
.meeting-type-wrap,
.meeting-subject-wrap,
.notes-wrap {
  font-size: 11px;
```

```
  color: #666;
  letter-spacing: 0.33px;
  height: 24px;
  padding: 5px;
}
.event-content div label {
  display: inline-block;
  min-width: 45px;
  color: #666;
}
.event-content div span {
  font-size: 11px;
  color: #151515;
  letter-spacing: 0.33px;
  line-height: 14px;
  padding-left: 8px;
}
.cell-footer.e-btn {
  background-color: #ffffff;
  border-color: #878787;
  color: #878787;
}
.cell-footer {
  padding-top: 10px;
}
.e-quick-popup-wrapper .e-cell-popup .e-popup-content {
  padding: 0 14px;
}
.e-quick-popup-wrapper .e-event-popup .e-popup-footer {
  display: block;
}
.e-quick-popup-wrapper .e-popup-footer button:first-child {
  margin-right: 5px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/quick-info-cs2" %}

Enable scroll option on all day section in Vue Schedule component

When you have larger number of appointments in all-day row, it is difficult to view all the appointments properly. In that case you can enable scroller option for all-day row by setting true to `enableAllDayScroll` whereas its default value is false. When setting this property to true, individual scroller for all-day row is enabled when it reaches its maximum height on expanding.

Note: This property is not applicable for Scheduler with height `auto`.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <ejs-schedule :height='height' :width='width'
:selectedDate='selectedDate'  :eventSettings='eventSettings'
:enableAllDayScroll='enableAllDayScroll'></ejs-schedule>
    </div>
  </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda, DragAndDrop }
from '@syncfusion/ej2-vue-schedule';
import { generateObject } from './datasource.js';
Vue.use(SchedulePlugin);
export default {
  data (){
    return {
      height: '550px',
      width: '100%',
      eventSettings: { dataSource: generateObject() },
      selectedDate: new Date(2021, 3, 28),
      enableAllDayScroll: true
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda, DragAndDrop]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/virtual-scrolling-cs1" %}

Two way binding in Vue Schedule component

We can achieve two-way binding with Vue Scheduler by using the v-model directive in Vue. The following Vue Scheduler properties support two-way binding.

- selectedDate
- currentView

Two-Way Binding with Vue Scheduler Component's selectedDate Property

In the following example, when you change the selectedDate value in the Vue Scheduler component, v-model will automatically update the value in the Date picker component, and if you change the value in the Date picker component, it will automatically update the value in the Vue Scheduler.

Copyright © 2001 -2024 Syncfusion Inc.

The following example demonstrates how to set the two-way-binding with the selectedDate property in the Vue Scheduler.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <div class='date-picker-container'>
            <ejs-datepicker id='date-picker' placeholder='Selected date'
floatLabelType="Always"  :value='selectedDate' :showClearButton='false' v-
model='selectedDate'>
            </ejs-datepicker>
        </div>
        <div class="schedule-container">
            <ejs-schedule :height='height' :selectedDate='selectedDate' v-
model='selectedDate' :eventSettings='eventSettings'></ejs-schedule>
        </div>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { DatePickerPlugin } from '@syncfusion/ej2-vue-calendars';
let currentYear = new Date().getFullYear();
let data = [{
    Id: 1,
    Subject: 'Paris',
    StartTime: new Date(currentYear, 1, 15, 10, 0),
    EndTime: new Date(currentYear, 1, 15, 12, 30),
    IsAllDay: false,
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=5'
}];
Vue.use(SchedulePlugin);
Vue.use(DatePickerPlugin);
export default {
  data (){
    return {
      height: '550px',
      views: ['Day', 'Week', 'WorkWeek', 'Month', 'Agenda'],
      eventSettings: {
        dataSource: data
      },
      selectedDate: new Date(currentYear, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
  .date-picker-container {
    width: 20%;
    margin: auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/two-way-binding-cs1" %}

Two-Way Binding with Vue Scheduler Component's currentView Property

In the following example, when you change the currentDate value in the Vue Scheduler component, v-model will automatically update the value in the Dropdown component, and if you change the value in the Dropdown component, it will automatically update the value in the Vue Scheduler.

The following example demonstrates how to set the two-way-binding with the currentView property in the Vue Scheduler.

APP.VUE

```
<template>
  <div id='app'>
    <div id='container'>
        <div class='drop-down-container'>
            <ejs-dropdownlist id='current-view-drop-down'
placeholder="Current view" floatLabelType="Always"  :dataSource='views'
:value='currentView' v-model="currentView">
            </ejs-dropdownlist>
        </div>
        <div class="schedule-container">
            <ejs-schedule :height='height' :selectedDate='selectedDate'
:currentView='currentView' v-model='currentView'
:eventSettings='eventSettings'></ejs-schedule>
        </div>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SchedulePlugin, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-vue-schedule';
import { DropDownListPlugin } from '@syncfusion/ej2-vue-dropdowns';
let currentYear = new Date().getFullYear();
let data = [{
    Id: 1,
    Subject: 'Paris',
```

```
    StartTime: new Date(currentYear, 1, 15, 10, 0),
    EndTime: new Date(currentYear, 1, 15, 12, 30),
    IsAllDay: false,
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=5'
}];
Vue.use(SchedulePlugin);
Vue.use(DropDownListPlugin);
export default {
  data (){
    return {
      height: '550px',
      currentView: 'Week',
      views: ['Day', 'Week', 'WorkWeek', 'Month', 'Agenda'],
      eventSettings: {
        dataSource: data
      },
      selectedDate: new Date(currentYear, 1, 15),
    }
  },
  provide: {
    schedule: [Day, Week, WorkWeek, Month, Agenda]
  }
}
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
  .drop-down-container {
    width: 20%;
    margin: auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/two-way-binding-cs2" %}

You can refer to our Vue Scheduler feature tour page for its groundbreaking feature representations. You can also explore our Vue Scheduler example to knows how to present and manipulate data.

Manual refresh in Vue Schedule component

In Scheduler, we can able to refresh the layout manually without re-render the DOM element by using the refreshLayout public method. The following example code explains to know how to use the refreshLayout method.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div>
        <div id='app'>
            <div id='container'>
                <tr>
                    <td>
                        <div>
                            <ejs-button cssClass='e-info' v-
on:click.native='onRefreshLayout'>Refresh Layout</ejs-button>
                        </div>
                    </td>
                </tr><br>
                <ejs-schedule id='Schedule' width='100%' height='485px'
:eventSettings='eventSettings' :selectedDate='selectedDate'>
                    <e-views>
                        <e-view option='Day'></e-view>
                        <e-view option='Week'></e-view>
                        <e-view option='WorkWeek'></e-view>
                        <e-view option='Month'></e-view>
                    </e-views>
                </ejs-schedule>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { extend } from '@syncfusion/ej2-base';
    import { SchedulePlugin, Day, Week, WorkWeek, Month } from
'@syncfusion/ej2-vue-schedule';
    import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
    Vue.use(ButtonPlugin);
    Vue.use(SchedulePlugin);
    export default {
        data () {
            return {
                eventSettings: {
                    dataSource:  [{
                        Id: 1,
                        Subject: 'Testing',
                        StartTime: new Date(2021, 10, 16, 10, 0),
                        EndTime: new Date(2021, 10, 16, 12, 0),
                        IsAllDay: false
                    }, {
                        Id: 2,
                        Subject: 'Vacation',
                        StartTime: new Date(2021, 10, 18, 10, 0),
                        EndTime: new Date(2021, 10, 18, 12, 0),
                        IsAllDay: false
                    }]
                },
                selectedDate: new Date(2021, 10, 15)
            }
        },
        provide: {
```

```
            schedule: [Day, Week, WorkWeek, Month]
        },
        methods: {
            onRefreshLayout: function () {
                let scheduleObj =
document.getElementById('Schedule').ej2_instances[0];
                scheduleObj.refreshLayout();
            }
        }
    }
</script>
<style>
  @import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
popups/styles/material.css';
  @import '../../node_modules/@syncfusion/ej2-vue-
schedule/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/schedule/app-crud-cs3" %}

Sidebar

Getting Started with the Vue Sidebar Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Sidebar component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies are required to use the Sidebar component in your application.

`js

|-- @syncfusion/ej2-vue-navigations

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-data

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-navigations

|-- @syncfusion/ej2-inputs

Copyright © 2001 -2024 Syncfusion Inc.

|-- @syncfusion/ej2-splitbuttons

|-- @syncfusion/ej2-lists

|-- @syncfusion/ej2-popups

|-- @syncfusion/ej2-buttons

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

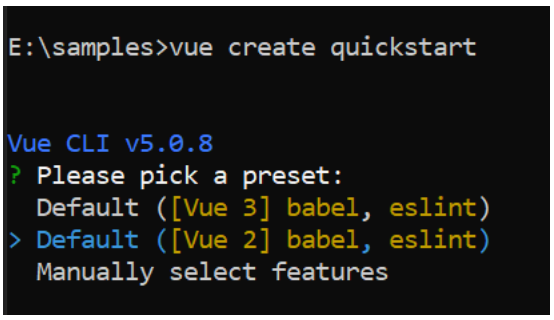
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

Copyright © 2001 -2024 Syncfusion Inc.

This article uses the Vue Sidebar component as an example. Install the @syncfusion/ej2-vue-navigations package by running the following command:

`bash

npm install @syncfusion/ej2-vue-navigations --save

`

or

`bash

yarn add @syncfusion/ej2-vue-navigations

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Sidebar component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Note: If you want to refer the combined component styles, please make use of our CRG (Custom Resource Generator) in your application.

Add Syncfusion Vue component

Follow the below steps to add the Vue Sidebar component using Composition API or Options API:

1\. First, import and register the Sidebar component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
components: {
'ejs-sidebar': SidebarComponent
}
}
</script>
```

2\. In the template section, define the Sidebar component.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-sidebar>
<div class="title"> Sidebar content</div>
</ejs-sidebar>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
        <div class="wrapper">
            <ejs-sidebar id="default-sidebar">
                <div class="title"> Sidebar content</div>
            </ejs-sidebar>
            <div>
                <div class="title">Main content</div>
                <div class="sub-title">Content goes here.</div>
            </div>
        </div>
    </div>
</template>
<script setup>
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    color: #ffffff;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
        <div class="wrapper">
            <ejs-sidebar id="default-sidebar">
                <div class="title"> Sidebar content</div>
            </ejs-sidebar>
            <div>
                <div class="title">Main content</div>
                <div class="sub-title">Content goes here.</div>
            </div>
        </div>
    </div>
</template>
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
export default {
    components: {
        'ejs-sidebar': SidebarComponent
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/sidebar/getting-started-cs4" %}

Enable backdrop

Enabling the showBackdrop in the Sidebar component will prevent the main content from user interactions, when it is in expanded state. Here, the DOM elements will not get changed. It only closes the main content by covering with a black backdrop overlay and focuses the Sidebar in the screen. Sidebar can be rendered with specific width by setting `width` property.

Note: To achieve a proper **backdrop**, we suggest that you create a wrapper parent container for the div block in which you intend to enable the backdrop. Set the class name of this parent container as the **target** for the Sidebar. Alternatively, you can place an empty div container after the target container.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
    <div class="wrapper">
        <ejs-sidebar id="default-sidebar" ref="sidebar"
:showBackdrop="showBackdrop" :closeOnDocumentClick="closeOnDocumentClick">
           <div class="title"> Sidebar content</div>
            <div class="sub-title">
               Click the button to close the Sidebar.
            </div>
            <div class="center-align">
               <button  id="close" v-on:click="closeClick"  class="e-btn
close-btn">Close Sidebar</button>
            </div>
        </ejs-sidebar>
        <div>
            <div class="title">Main content</div>
                <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
                   <div style="padding:20px" class="center-align">
                       <button id="toggle"  class="e-btn e-info" v-
on:click="toggleClick" >Toggle Sidebar</button>
                   </div>
           </div>
       </div>
    </div>
</template>
<script setup>
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
import { ref } from "vue";
const showBackdrop  = true;
const closeOnDocumentClick = true;
const sidebar = ref(null);
const toggleClick = () => {
  sidebar.value.toggle();
```

```
}
const closeClick = () => {
  sidebar.value.hide();
}

</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.center-align {
    text-align: center;
    padding: 20px;
}
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
.close-btn:hover {
    color: #fafafa;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
    <div class="wrapper">
        <ejs-sidebar id="default-sidebar" ref="sidebar"
:showBackdrop="showBackdrop" :closeOnDocumentClick="closeOnDocumentClick">
           <div class="title"> Sidebar content</div>
            <div class="sub-title">
               Click the button to close the Sidebar.
            </div>
            <div class="center-align">
               <button  ejs-button  id="close" v-on:click="closeClick"
class="e-btn close-btn">Close Sidebar</button>
            </div>
```

```
        </ejs-sidebar>
        <div>
            <div class="title">Main content</div>
                <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
                    <div style="padding:20px" class="center-align">
                        <button ejs-button id="toggle"  class="e-btn e-info"
v-on:click="toggleClick" >Toggle Sidebar</button>
                    </div>
            </div>
       </div>
    </div>
</template>
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
import { ButtonComponent } from '@syncfusion/ej2-vue-buttons';
export default {
     components: {
        'ejs-sidebar': SidebarComponent
        'ejs-button': ButtonComponent
     },
     data () {
        return {
          showBackdrop :true,
          closeOnDocumentClick:true,
        }
    },
    methods: {
        toggleClick :function() {
          this.$refs.sidebar.toggle();
        },
        closeClick: function() {
         this.$refs.sidebar.hide();
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.center-align {
    text-align: center;
    padding: 20px;
}
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
.center {
```

```
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
.close-btn:hover {
    color: #fafafa;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/backdrop-cs1" %}

Position

Positioning the Sidebar to the right or left of the main content can be achieved by using the position property. If the position is not set, the Sidebar will expand from the left to the body element. enablePersistence will persist the component's state between page reloads. change event will be triggered when the state(expand/collapse) of the component is changed.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
    <ejs-sidebar id="default-sidebar" ref="sidebar" :type="type"
:target="target" :position="position"
:enablePersistence="enablePersistence">
      <div class="title"> Sidebar content</div>
       <div class="sub-title">
        Click the button to close the Sidebar.
    </div>
    <div class="center-align">
        <button ejs-button id="close" v-on:click="closeClick" class="e-btn
close-btn">Close Sidebar</button>
    </div>
</ejs-sidebar>
<div id="head">
    <ejs-button id="toggle" ref="togglebtn" class="e-btn e-info"
cssClass="e-flat" iconCss="e-icons burg-icon" isToggle="true" v-
on:click.native="btnClick">Open</ejs-button>
</div>
<div>
<div id="maincontent" class="content">
    <div>
        <div class="title">Main content</div>
            <div class="rows">
                <div class="row">
                  <ejs-radiobutton id="left" label='Left' name='state'
checked='true' :change="positionChange"></ejs-radiobutton>
                </div>
                <div class="row">
                  <ejs-radiobutton id="right" label='Right' name='state'
:change="positionChange"></ejs-radiobutton>
```

```
                </div>
            </div>
        </div>
        </div>
</div>
</div>
<!--end of main content declaration -->
</template>
<script setup>
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
import { ButtonComponent as EjsButton ,RadioButtonComponent as
EjsRadiobutton } from '@syncfusion/ej2-vue-buttons';
import { ref } from "vue";
const type = 'Push';
const target = '.content';
const position = ref('Left');
const enablePersistence = true;
const sidebar = ref(null);
const togglebtn = ref('Open');

const positionChange = (args) => {
        position.value = args.event.target.id == "left" ? "Left" : "Right";
        }
const btnClick = () => {
        if(togglebtn.value.$el.classList.contains('e-active')){
            togglebtn.value.Content = 'Open';
            sidebar.value.hide();
        }
        else{
            togglebtn.value.Content = 'Close';
            sidebar.value.show();
        }
    };
const closeClick = () => {
         sidebar.value.hide();
         togglebtn.value.$el.classList.remove('e-active');
         togglebtn.value.Content = 'Open';
    };

</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.rows{
    margin:auto;
    text-align:center;
}
.row{
    display:inline-block;
    padding:10px;
    margin:auto;
}
.center-align {
    text-align: center;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    padding: 20px;
}
.burg-icon:before{
    content: '\e10d';
    font-size:16px;
}
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
#head {
    border: 1px solid #424242;
    border-bottom-color:transparent;
    background:#00897B;
}
#toggle,#container .e-btn.e-info:hover,#toggle:focus { /* csslint allow:
adjoining-classes*/
    background:#00695C;
    box-shadow:none;
    border-radius:0;
    height:39px;
    width:100px;
}
#close,#close:hover,#close:active,#close:focus{ /* csslint allow: adjoining-
classes*/
    background: #fafafa;
    color:black
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
.radiobutton{
    display:inline-block;
    padding:10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
#default-sidebar {
    background-color: #26A69A;
    color: #ffffff;
}
.close-btn:hover {
    color: #fafafa;
}
#toggle {
   color:white;
}
.content{
    height:305px;
```

```
    border: 1px solid grey;
}
@font-face {
    font-family: 'e-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAkAAAAg4aGVhZBLt+DYAAADQAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAAYAAAAA
wbG9jYQukCgIAAAIkAAAAGm1heHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAzoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAAAAADAABAAAAAQAAlbrm7l8
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAD8wPzAAAACAACAAAAAAAAAAAEAAAAMAQIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAAgAAAAMAAAAUAAMAAQA
AABQABABeAAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAgABwAKAAkABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAAAAAIAAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAA+oD6gALAAATCQEXCQEHCQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAD6gOkAAMABwALAAA3IRUhESEVIREhFSE
VA9b8KgPW/CoD1vwq6I0B64wB640AAAEAAAAA+oD4QCaAAABMx8aHQEPDjEPAh8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBwYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHBoZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAwIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICAkJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBwgICAkKCgs
LDAwNDQ4ODxAQEBEWFxYWFhYVFRQUExIRERAOFxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEhERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTExUUFhUWFxYWFxEQEBAPDg4
NDQwMCwsKCgkICAgHBgYFBQQEBQQAAAAAAwAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8fFz8XLxcPBgI+BQQDAwMCAT8EBAM
DAwIBAQIDAwMEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAwMDAgEBAgMDAwQE0AUEAwMDAgEBAgMDAwQ
FfAUEAwMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEB0AECAwMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAwMDAgEBAgMDAwQFaAQEAwQCAgElERMVFhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxYVExEQDgsJCAUDAQEDBQcKCw4QERMVFhcZGxsdHh4fICAhIiIhICAfHh4dHBoZFxYVExE
QDgsKBwUDAQEDBQcKCw4AAAIAAAAAA9MD6QALAE8AAAEOAQcuASc+ATceAQEHBgcnJgYPAQYWHwE
GFBcHDgEfAR4BPwEWHwEeATsBMjY/ATY3FxY2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGAps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAwdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYMhhSBhIIrgg
FAzIfE4QJDAwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOECQwMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4zA+v+Ff4VHwHMAc0AAAAAAQAAAAADAQPqAAUAAAEXCQEHAQLlHf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAD8wPzAEAAgQEBAAAlDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUfHTsBPx09AS8dKwEPHQL1DQ0ODg4PDw8QEBAQERERERUUFBQTExITERERERBAPDw0ODAw
LCwkJCAcGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAgQDBQUGBgcHCQgJCv3QDQ0ODg4PDw8QEBA
QERERERUUFBQTExITERERERBAPDw0ODAwLCwkJCAcGBgQEAgL8fgIDBQUHCAkKCwwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcYFhYVFRQUExIREQ8PDg0MCwoJCAcFBQMCAgMFBQcICQoLDA0ODw8
RERITFBQVFhYWGBcYGRkZGhoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETEhMTFBQUFRERERERQEBAQDw8PDg4ODQ31ERE
RERAQEBAPDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETEhM
TFBQUFRoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBU
WFhYYFxgZGRkaGhkZGRgXGBYWFhUUFBMSEREPDw4NDAsKCQgHBQUDAgIDBQUHCAkKCwwNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgwMDAsKCggIBwU
FAwIBAQIDBQUHCAgKCgsMDAwODf00DQ4MDAwLCgoICAcFBQMCAQECAwUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgsLDA0NDv00Dg0NDAsLCgoICAcFBQMCAQECAwUFBwgICgoLCwwNDQ4CzA4NDQwLCwo
KCAgHBQUDAgAAABIA3gABAAAAAAAAAAEAAAABAAAAAAAABAA0AAQABAAAAAAACAAcADgABAAAAAAA
DAA0AFQABAAAAAAAAEAA0AIgABAAAAAAAAFAAsALwABAAAAAAAGAA0AOgABAAAAAAAKACwARwABAAA
AAAALABIAcwADAAEECQAAAAIAhQADAAEECQABABoAhwADAAEECQACAA4AoQADAAEECQADABoArwA
DAAEECQAEABoAyQADAAEECQAFABYA4wADAAEECQAGABoA+QADAAEECQAKAFgBEwADAAEECQALACQ
BayBlLWljb25zLW1ldHJvUmVndWxhcmUtaWNvbnMtbWV0cm9lLWljb25zLW1ldHJvVmVyc2lvbiA
```

```
xLjBlLWljb25zLW1ldHJvRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwBlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQByAHMAaQBvAG4AIAAxAC4AMABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAB
NAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwBAgEDAQQBBQEGAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMHbWVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHQMY2hldnJvbi1sZWZ0CE1UX0NsZWFyDE1UX0p1bmttYWlscwR
zdG9wAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
    <ejs-sidebar id="default-sidebar" ref="sidebar" :type="type"
:target="target" :position="position"
:enablePersistence="enablePersistence">
      <div class="title"> Sidebar content</div>
       <div class="sub-title">
        Click the button to close the Sidebar.
    </div>
    <div class="center-align">
        <button ejs-button id="close" v-on:click="closeClick" class="e-btn
close-btn">Close Sidebar</button>
    </div>
</ejs-sidebar>
<div id="head">
    <ejs-button id="toggle" ref="togglebtn" class="e-btn e-info"
cssClass="e-flat" iconCss="e-icons burg-icon" isToggle="true" v-
on:click.native="btnClick">Open</ejs-button>
</div>
<div>
<div id="maincontent" class="content">
    <div>
        <div class="title">Main content</div>
            <div class="rows">
                <div class="row">
                  <ejs-radiobutton id="left" label='Left' name='state'
checked='true' :change="positionChange"></ejs-radiobutton>
                </div>
                <div class="row">
                  <ejs-radiobutton id="right" label='Right' name='state'
:change="positionChange"></ejs-radiobutton>
                </div>
            </div>
        </div>
        </div>
</div>
</div>
<!--end of main content declaration -->
</template>
```

```
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
import { ButtonComponent ,RadioButtonComponent } from '@syncfusion/ej2-vue-
buttons';
export default {
    components: {
      'ejs-sidebar': SidebarComponent,
      'ejs-button': ButtonComponent,
      'ejs-radiobutton': RadioButtonComponent
    },
    data () {
        return {
         type :'Push',
         target : '.content',
         position : 'Left',
         enablePersistence: true
        }
    },
    methods: {
        positionChange:function(args) {
          this.position = args.event.target.id == "left" ? "Left" : "Right";
        },
        btnClick: function(){
        if(this.$refs.togglebtn.$el.classList.contains('e-active')){
            this.$refs.togglebtn.Content = 'Open';
            this.$refs.sidebar.hide();
        }
        else{
            this.$refs.togglebtn.Content = 'Close';
            this.$refs.sidebar.show();
        }
    },
        closeClick: function() {
         this.$refs.sidebar.hide();
         this.$refs.togglebtn.$el.classList.remove('e-active');
         this.$refs.togglebtn.Content = 'Open';
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.rows{
    margin:auto;
    text-align:center;
}
.row{
    display:inline-block;
    padding:10px;
    margin:auto;
}
.center-align {
    text-align: center;
    padding: 20px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
.burg-icon:before{
    content: '\e10d';
    font-size:16px;
}
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
#head {
    border: 1px solid #424242;
    border-bottom-color:transparent;
    background:#00897B;
}
#toggle,#container .e-btn.e-info:hover,#toggle:focus { /* csslint allow:
adjoining-classes*/
    background:#00695C;
    box-shadow:none;
    border-radius:0;
    height:39px;
    width:100px;
}
#close,#close:hover,#close:active,#close:focus{ /* csslint allow: adjoining-
classes*/
    background: #fafafa;
    color:black
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
.radiobutton{
    display:inline-block;
    padding:10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
#default-sidebar {
    background-color: #26A69A;
    color: #ffffff;
}
.close-btn:hover {
    color: #fafafa;
}
#toggle {
   color:white;
}
.content{
    height:305px;
    border: 1px solid grey;
```

```
}
@font-face {
    font-family: 'e-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAkAAAAg4aGVhZBLt+DYAAADQAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAAYAAAAA
wbG9jYQukCgIAAAIkAAAAGm1heHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAzoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAAAAAADAABAAAAAQAAlbrm7l8
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAD8wPzAAAACAACAAAAAAAAAAAEAAAAMAQIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAAgAAAAMAAAAUAAMAAQA
AABQABABeAAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAgABwAKAAkABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAAAAAIAAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAA+oD6gALAAATCQEXCQEHCQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAD6gOkAAMABwALAAA3IRUhESEVIREhFSE
VA9b8KgPW/CoD1vwq6I0B64wB640AAAEAAAAAA+oD4QCaAAABMx8aHQEPDjEPAh8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBwYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHBoZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAwIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICAkJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBwgICAkKCgs
LDAwNDQ4ODxAQEBEWFxYWFhYVFRQUExIRERAOFxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEhERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTExUUFhUWFxYWFxEQEBAPDg4
NDQwMCwsKCgkICAgHBgYFBQQEBQQAAAAAAwAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8fFz8XLxcPBgI+BQQDAwMCAT8EBAM
DAwIBAQIDAwMEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAwMDAgEBAgMDAwQE0AUEAwMDAgEBAgMDAwQ
FfAUEAwMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEB0AECAwMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAwMDAgEBAgMDAwQFaAQEAwQCAgElERMVFhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxYVExEQDgsJCAUDAQEDBQcKCw4QERMVFhcZGxsdHh4fICAhIiIhICAfHh4dHBoZFxYVExE
QDgsKBwUDAQEDBQcKCw4AAAIAAAAAA9MD6QALAE8AAAEOAQcuASc+ATceAQEHBgcnJgYPAQYWHwE
GFBcHDgEfAR4BPwEWHwEeATsBMjY/ATY3FxY2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGAps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAwdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYMhhSBhIIrgg
FAzIfE4QJDAwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOECQwMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4zA+v+Ff4VHwHMAc0AAAAAQAAAAADAQPqAAUAAAEXCQEHAQLlHf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAAD8wPzAEAAgQEBAAAlDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUfHTsBPx09AS8dKwEPHQL1DQ0ODg4PDw8QEBAQERERERUUFBQTExITEREREBAPDw0ODAw
LCwkJCAcGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAgQDBQUGBgcHCQgJCv3QDQ0ODg4PDw8QEBA
QERERERUUFBQTExITEREREBAPDw0ODAwLCwkJCAcGBgQEAgL8fgIDBQUHCAkKCwwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcYFhYWFRQUExIREQ8PDg0MCwoJCAcFBQMCAgMFBQcICQoLDA0ODw8
RERITFBQVFhYWGBcYGRkZGhoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBAREREUEhMTFBQUFREREREQEBAQDw8PDg4ODQ31ERE
RERAQEBAPDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBAREREUEhM
TFBQUFRoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBU
WFhYYFxgZGRkaGhkZGRgXGBYWFhUUFBMSEREPDw4NDAsKCQgHBQUDAgIDBQUHCAkKCwwNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgwMDAsKCggIBwU
FAwIBAQIDBQUHCAgKCgsMDAwODf00DQ4MDAwLCgoICAcFBQMCAQECAwUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgsLDA0NDv00Dg0NDAsLCgoICAcFBQMCAQECAwUFBwgICgoLCwwNDQ4CzA4NDQwLCwo
KCAgHBQUDAgAAABIA3gABAAAAAAAAAAEAAAABAAAAAAAABAA0AAQABAAAAAAACAAcADgABAAAAAAA
DAA0AFQABAAAAAAAAEAA0AIgABAAAAAAAAFAAsALwABAAAAAAAAGAA0AOgABAAAAAAAKACwARwABAAA
AAAALABIAcwADAAEECQAAAAIAhQADAAEECQABABoAhwADAAEECQACAA4AoQADAAEECQADABoArwA
DAAEECQAEABoAyQADAAEECQAFABYA4wADAAEECQAGABoA+QADAAEECQAKAFgBEwADAAEECQALACQ
BayBlLWljb25zLW1ldHJvUmVndWxhcmUtaWNvbnMtbWV0cm9lLWljb25zLW1ldHJvVmVyc2lvbiA
xLjBlLWljb25zLW1ldHJvRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
```

```
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwBlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQByAHMAaQBvAG4AIAAxAC4AMABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAB
NAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwBAgEDAQQBBQEGAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMHbWVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHQMY2hldnJvbi1sZWZ0CE1UX0NsZWFyDE1UX0p1bmttYWlscwR
zdG9wAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/position-cs2" %}

Animate

Animation transitions can be set while expanding or collapsing the Sidebar using the animate property. By default , animate property is set to true. enableRTL will display the sidebar in the right-to-left direction.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
    <div class="wrapper">
        <ejs-sidebar id="default-sidebar" ref="sidebar" :type="type"
:width="width" :animate="animate" :enableRtl="enableRtl">
           <div class="title"> Sidebar content</div>
            <div class="sub-title">
               Click the button to close the Sidebar
            </div>
            <div class="center-align">
               <button  ejs-button  id="close" v-on:click="closeClick"
class="e-btn close-btn">Close Sidebar</button>
            </div>
        </ejs-sidebar>
        <div>
            <div class="title">Main content</div>
                <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
                    <div style="padding:20px" class="center-align">
                        <button ejs-button id="toggle"  class="e-btn e-info"
v-on:click="toggleClick" >Toggle Sidebar</button>
                    </div>
           </div>
       </div>
    </div>
</template>
<script setup>
import { ref } from 'vue';
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
const animate = false;
const enableRtl = true;
const width = '280px';
```

```
const type = 'Push';
const sidebar = ref(null);
const  toggleClick = () => {
    sidebar.value.toggle();
};
const closeClick = () => {
    sidebar.value.hide();
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.center-align {
    text-align: center;
    padding: 20px;
}
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
.close-btn:hover {
    color: #fafafa;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
    <div class="wrapper">
        <ejs-sidebar id="default-sidebar" ref="sidebar" :type="type"
:width="width" :animate="animate" :enableRtl="enableRtl">
            <div class="title"> Sidebar content</div>
             <div class="sub-title">
                Click the button to close the Sidebar
             </div>
             <div class="center-align">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <button  ejs-button  id="close" v-on:click="closeClick"
class="e-btn close-btn">Close Sidebar</button>
            </div>
        </ejs-sidebar>
        <div>
            <div class="title">Main content</div>
                <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
                    <div style="padding:20px" class="center-align">
                        <button ejs-button id="toggle"  class="e-btn e-info"
v-on:click="toggleClick" >Toggle Sidebar</button>
                    </div>
            </div>
        </div>
    </div>
</template>
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
export default {
     components: {
        'ejs-sidebar': SidebarComponent
     },
     data () {
        return {
            animate:false,
            enableRtl: true,
            width:'280px',
            type:'Push'
        }
    },
    methods: {
        toggleClick :function() {
          this.$refs.sidebar.toggle();
        },
        closeClick: function() {
         this.$refs.sidebar.hide();
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.center-align {
    text-align: center;
    padding: 20px;
}
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 16px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    padding: 10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
.close-btn:hover {
    color: #fafafa;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/animate-cs1" %}

Close on document click

Sidebar can be closed on document click by setting closeOnDocumentClick to true. If this property is not set, the Sidebar will not close on document click since its default value is false. Sidebar can be kept opened during rendering using isOpen property.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
    <div class="wrapper">
        <ejs-sidebar id="default-sidebar" ref="sidebar" :width="width"
:type="type" :isOpen="isOpen" :closeOnDocumentClick="closeOnDocumentClick">
           <div class="title"> Sidebar content</div>
        </ejs-sidebar>
        <div>
            <div class="title">Main content</div>
                <div class="sub-title"> Click the button to open the
Sidebar.</div>
                   <div style="padding:20px" class="center-align">
                       <button ejs-button id="toggle"  class="e-btn e-info"
v-on:click="toggleClick" >Open Sidebar</button>
                </div>
           </div>
       </div>
    </div>
</template>
<script setup>
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
import { ref } from 'vue';
const closeOnDocumentClick = true;
const isOpen = true;
const type = 'Push';
const width = '280px';
const sidebar = ref(null);
const toggleClick = () => {
```

```
        sidebar.value.toggle();
        };
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.center-align {
    text-align: center;
    padding: 20px;
}
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
.close-btn:hover {
    color: #fafafa;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
    <div class="wrapper">
        <ejs-sidebar id="default-sidebar" ref="sidebar" :width="width"
:type="type" :isOpen="isOpen" :closeOnDocumentClick="closeOnDocumentClick">
           <div class="title"> Sidebar content</div>
        </ejs-sidebar>
        <div>
            <div class="title">Main content</div>
                <div class="sub-title"> Click the button to open the
Sidebar.</div>
                    <div style="padding:20px" class="center-align">
                        <button ejs-button id="toggle"  class="e-btn e-info"
v-on:click="toggleClick" >Open Sidebar</button>
                </div>
           </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </div>
    </div>
</template>
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
export default {
     components: {
        'ejs-sidebar': SidebarComponent
     },
     data () {
        return {
            closeOnDocumentClick: true,
            isOpen:true,
            type:'Push',
            width:'280px'
        }
    },
    methods: {
        toggleClick :function() {
          this.$refs.sidebar.toggle();
        },
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.center-align {
    text-align: center;
    padding: 20px;
}
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
.close-btn:hover {
    color: #fafafa;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/document-click-cs1" %}

Enable gestures

Expand or collapse the Sidebar while swiping in touch devices using `enableGestures` property. By default, `enableGestures` is set to true.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
    <div class="wrapper">
        <ejs-sidebar id="default-sidebar" ref="sidebar" :width="width"
:type="type" :enableGestures="enableGestures">
            <div class="title"> Sidebar content</div>
             <div class="sub-title">
                Click the button to close the Sidebar.
             </div>
             <div class="center-align">
                <button  ejs-button  id="close" v-on:click="closeClick"
class="e-btn close-btn">Close Sidebar</button>
             </div>
        </ejs-sidebar>
        <div>
            <div class="title">Main content</div>
                <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
                    <div style="padding:20px" class="center-align">
                        <button ejs-button id="toggle"  class="e-btn e-info"
v-on:click="toggleClick" >Toggle Sidebar</button>
                    </div>
            </div>
       </div>
    </div>
</template>
<script setup>
import { ref } from 'vue';
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
const enableGestures = false;
const type = 'Push';
const width = '280px';
const sidebar = ref(null);
const toggleClick = () => {
    sidebar.value.toggle();
};
const closeClick = () => {
    sidebar.value.hide();
};

</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.center-align {
    text-align: center;
    padding: 20px;
}
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
.close-btn:hover {
    color: #fafafa;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
    <div class="wrapper">
        <ejs-sidebar id="default-sidebar" ref="sidebar" :width="width"
:type="type" :enableGestures="enableGestures">
           <div class="title"> Sidebar content</div>
            <div class="sub-title">
               Click the button to close the Sidebar.
            </div>
            <div class="center-align">
               <button  ejs-button  id="close" v-on:click="closeClick"
class="e-btn close-btn">Close Sidebar</button>
            </div>
        </ejs-sidebar>
        <div>
            <div class="title">Main content</div>
                <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
                   <div style="padding:20px" class="center-align">
                       <button ejs-button id="toggle"  class="e-btn e-info"
v-on:click="toggleClick" >Toggle Sidebar</button>
                </div>
```

```
            </div>
        </div>
    </div>
</template>
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
export default {
     components: {
        'ejs-sidebar': SidebarComponent
     },
     data () {
        return {
            enableGestures: false,
            type:'Push',
            width:'280px'
        }
    },
    methods: {
        toggleClick :function() {
            this.$refs.sidebar.toggle();
        },
        closeClick: function() {
            this.$refs.sidebar.hide();
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.center-align {
    text-align: center;
    padding: 20px;
}
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.close-btn:hover {
    color: #fafafa;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/gestures-cs1" %}

See Also

- Sidebar with navigation menu
- Sidebar responsive panel
- Sidebar with listView

Getting Started with the Vue Sidebar Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Sidebar component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

Copyright © 2001 -2024 Syncfusion Inc.

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Copyright © 2001 -2024 Syncfusion Inc.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Sidebar component as an example. To use the Vue Sidebar component in the project, the @syncfusion/ej2-vue-navigations package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-navigations --save

`

or

`bash

yarn add @syncfusion/ej2-vue-navigations

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Sidebar component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Sidebar component using Composition API or Options API:

1.First, import and register the Sidebar component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
//Component registration
export default {
name: "App",
components: {
"ejs-sidebar": SidebarComponent
}
}
</script>
```

2.Add the component definition in template section.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-sidebar id="default-sidebar">
<div class="title"> Sidebar content</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
<div class="sub-title">Content goes here.</div>
</div>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-sidebar id="default-sidebar">
<div class="title"> Sidebar content</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
<div class="sub-title">Content goes here.</div>
</div>
</div>
</template>
<script setup>
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
.title {
text-align: center;
font-size: 20px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
padding: 15px;
}
.sub-title {
text-align: center;
font-size: 16px;
padding: 10px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-sidebar id="default-sidebar">
<div class="title"> Sidebar content</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
<div class="sub-title">Content goes here.</div>
</div>
</div>
</template>
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
//Component registration
export default {
name: "App",
components: {
"ejs-sidebar": SidebarComponent
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
.title {
text-align: center;
font-size: 20px;
padding: 15px;
}
.sub-title {
text-align: center;
font-size: 16px;
padding: 10px;
}
</style>
```

Run the project

To run the project, use the following command:

Copyright © 2001 -2024 Syncfusion Inc.

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

Enable backdrop

Enabling the showBackdrop in the Sidebar component will prevent the main content from user interactions, when it is in expanded state. Here, the DOM elements will not get changed. It only closes the main content by covering with a black backdrop overlay and focuses the Sidebar in the screen. Sidebar can be rendered with specific width by setting `width` property.

Note: To achieve a proper **backdrop**, we suggest that you create a wrapper parent container for the div block in which you intend to enable the backdrop. Set the class name of this parent container as the **target** for the Sidebar. Alternatively, you can place an empty div container after the target container.

In the following example, the `showBackdrop` and `closeOnDocumentClick` properties are enabled.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class="wrapper">
<ejs-sidebar id="default-sidebar" :showBackdrop="showBackdrop"
:closeOnDocumentClick="closeOnDocumentClick">
<div class="title"> Sidebar content</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
</div>
</div>
```

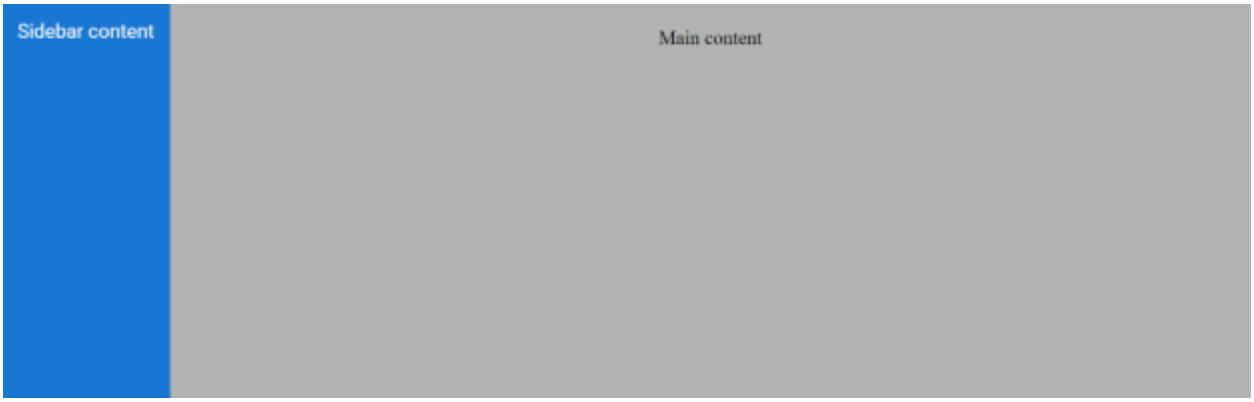
```
</div>
</template>
<script setup>
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
const showBackdrop= true;
const closeOnDocumentClick= true;
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.title {
text-align: center;
font-size: 20px;
padding: 15px;
}
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class="wrapper">
<ejs-sidebar id="default-sidebar" :showBackdrop="showBackdrop"
:closeOnDocumentClick="closeOnDocumentClick">
<div class="title"> Sidebar content</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
</div>
</div>
</div>
</template>
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
//Component registration
export default {
name: "App",
components: {
"ejs-sidebar": SidebarComponent
},
data() {
return {
showBackdrop: true,
closeOnDocumentClick: true,
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.title {
text-align: center;
font-size: 20px;
padding: 15px;
}
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
</style>
```



Position

Positioning the Sidebar to the right or left of the main content can be achieved by using the position property. If the position is not set, the Sidebar will expand from the left to the body element. enablePersistence will persist the component's state between page reloads. change event will be triggered when the state(expand/collapse) of the component is changed.

In the following example, the position property is set as Right.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class="wrapper">
<ejs-sidebar id="default-sidebar" :position="position">
<div class="title"> Sidebar content</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
</div>
</div>
</div>
</template>
<script setup>
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
const position= 'Right';
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

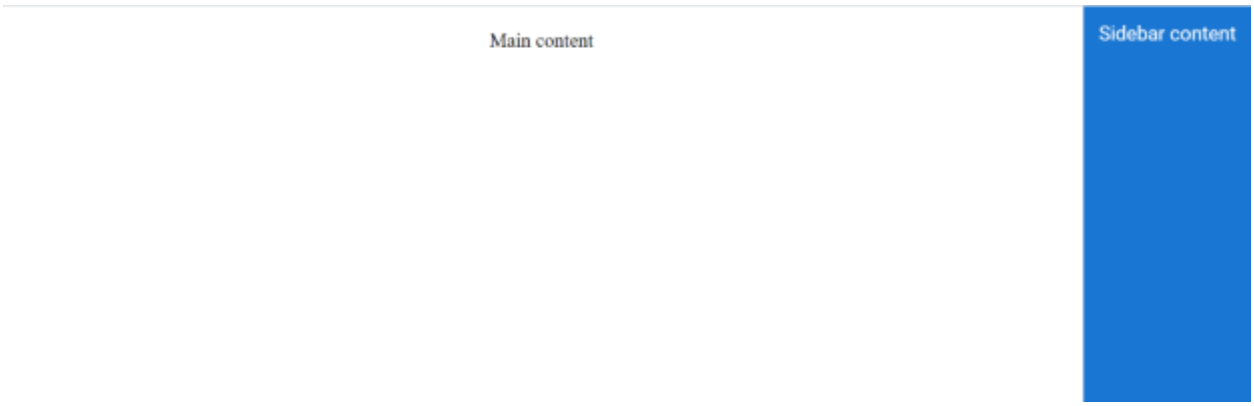
Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.title {
text-align: center;
font-size: 20px;
padding: 15px;
}
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class="wrapper">
<ejs-sidebar id="default-sidebar" :position="position">
<div class="title"> Sidebar content</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
</div>
</div>
</div>
</template>
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
//Component registration
export default {
name: "App",
components: {
"ejs-sidebar": SidebarComponent
},
data() {
return {
position: 'Right',
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.title {
text-align: center;
font-size: 20px;
padding: 15px;
}
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.



Animate

Animation transitions can be set while expanding or collapsing the Sidebar using the animate property. By default, animate property is set to true. enableRTL will display the sidebar in the right-to-left direction.

In the following example, the animate property is disabled.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class="wrapper">
<ejs-sidebar id="default-sidebar" ref="sidebar" :type="type" :width="width"
:animate="animate" :enableRtl="enableRtl">
<div class="title"> Sidebar content</div>
<div class="sub-title">
Click the button to close the Sidebar
</div>
<div class="center-align">
<ejs-button  id="close" v-on:click="closeClick"  class="e-btn close-
btn">Close Sidebar</ejs-button>
</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
<div class="sub-title"> Click the button to open/close the Sidebar.</div>
<div style="padding:20px" class="center-align">
<ejs-button id="toggle"  class="e-btn e-info" v-on:click="toggleClick"
>Toggle Sidebar</ejs-button>
</div>
</div>
</div>
</div>
</template>
<script setup>
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
import { ButtonComponent as EjsButton } from '@syncfusion/ej2-vue-buttons';
import { ref } from "vue";
const type = 'Push';
const width = '280px';
```

```
const animate = false;
const enableRtl = true;
const sidebar = ref(null);
const closeClick = () => {
sidebar.value.hide();
}
const toggleClick = () => {
sidebar.value.toggle();
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.center-align {
text-align: center;
padding: 20px;
}
.title {
text-align: center;
font-size: 20px;
padding: 15px;
}
.sub-title {
text-align: center;
font-size: 16px;
padding: 10px;
}
.center {
text-align: center;
display: none;
font-size: 13px;
font-weight: 400;
margin-top: 20px;
}
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
.close-btn:hover {
color: #fafafa;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class="wrapper">
<ejs-sidebar id="default-sidebar" ref="sidebar" :type="type" :width="width"
:animate="animate" :enableRtl="enableRtl">
<div class="title"> Sidebar content</div>
<div class="sub-title">
Click the button to close the Sidebar
</div>
```

```
<div class="center-align">
<ejs-button  id="close" v-on:click="closeClick"  class="e-btn close-
btn">Close Sidebar</ejs-button>
</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
<div class="sub-title"> Click the button to open/close the Sidebar.</div>
<div style="padding:20px" class="center-align">
<ejs-button id="toggle"  class="e-btn e-info" v-on:click="toggleClick"
>Toggle Sidebar</ejs-button>
</div>
</div>
</div>
</div>
</template>
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
import { ButtonComponent } from '@syncfusion/ej2-vue-buttons';
//Component registration
export default {
name: "App",
components: {
"ejs-sidebar": SidebarComponent,
"ejs-button": ButtonComponent
},
data() {
return {
type: 'Push',
width: '280px',
animate: false,
enableRtl: true,
}
},
methods: {
toggleClick: function () {
this.$refs.sidebar.toggle();
},
closeClick: function () {
this.$refs.sidebar.hide();
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.center-align {
text-align: center;
padding: 20px;
}
.title {
text-align: center;
font-size: 20px;
padding: 15px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
.sub-title {
text-align: center;
font-size: 16px;
padding: 10px;
}
.center {
text-align: center;
display: none;
font-size: 13px;
font-weight: 400;
margin-top: 20px;
}
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
.close-btn:hover {
color: #fafafa;
}
</style>
```

Close on document click

Sidebar can be closed on document click by setting closeOnDocumentClick to true. If this property is not set, the Sidebar will not close on document click since its default value is false. Sidebar can be kept opened during rendering using isOpen property.

In the following example, the closeOnDocumentClick property is enabled..

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class="wrapper">
<ejs-sidebar id="default-sidebar" ref="sidebar" :width="width" :type="type"
:isOpen="isOpen" :closeOnDocumentClick="closeOnDocumentClick">
<div class="title"> Sidebar content</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
```

```
<div class="sub-title"> Click the button to open the Sidebar.</div>
<div style="padding:20px" class="center-align">
<ejs-button id="toggle"  class="e-btn e-info" v-on:click="toggleClick" >Open
Sidebar</ejs-button>
</div>
</div>
</div>
</div>
</template>
<script setup>
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
import { ButtonComponent as EjsButton } from '@syncfusion/ej2-vue-buttons';
import { ref } from "vue";
const type = 'Push';
const width = '280px';
const closeOnDocumentClick = true;
const isOpen = true;
const sidebar = ref(null);
const toggleClick = () => {
sidebar.value.toggle();
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.center-align {
text-align: center;
padding: 20px;
}
.title {
text-align: center;
font-size: 20px;
padding: 15px;
}
.sub-title {
text-align: center;
font-size: 16px;
padding: 10px;
}
.center {
text-align: center;
display: none;
font-size: 13px;
font-weight: 400;
margin-top: 20px;
}
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
.close-btn:hover {
color: #fafafa;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class="wrapper">
<ejs-sidebar id="default-sidebar" ref="sidebar" :width="width" :type="type"
:isOpen="isOpen" :closeOnDocumentClick="closeOnDocumentClick">
<div class="title"> Sidebar content</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
<div class="sub-title"> Click the button to open the Sidebar.</div>
<div style="padding:20px" class="center-align">
<ejs-button id="toggle"  class="e-btn e-info" v-on:click="toggleClick" >Open
Sidebar</ejs-button>
</div>
</div>
</div>
</div>
</template>
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
import { ButtonComponent } from '@syncfusion/ej2-vue-buttons';
//Component registration
export default {
name: "App",
components: {
"ejs-sidebar": SidebarComponent,
"ejs-button": ButtonComponent
},
data() {
return {
type: 'Push',
width: '280px',
isOpen: false,
closeOnDocumentClick: true,
}
},
methods: {
toggleClick: function () {
this.$refs.sidebar.toggle();
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.center-align {
text-align: center;
padding: 20px;
}
.title {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
text-align: center;
font-size: 20px;
padding: 15px;
}
.sub-title {
text-align: center;
font-size: 16px;
padding: 10px;
}
.center {
text-align: center;
display: none;
font-size: 13px;
font-weight: 400;
margin-top: 20px;
}
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
.close-btn:hover {
color: #fafafa;
}
</style>
```

Enable gestures

Expand or collapse the Sidebar while swiping in touch devices using enableGestures property. By default, enableGestures is set to true.

In the following example, the enableGestures property is enabled.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class="wrapper">
<ejs-sidebar id="default-sidebar" ref="sidebar" :width="width" :type="type"
:enableGestures="enableGestures">
<div class="title"> Sidebar content</div>
<div class="sub-title">
```

```
Click the button to close the Sidebar.
</div>
<div class="center-align">
<ejs-button  id="close" v-on:click="closeClick"  class="e-btn close-
btn">Close Sidebar</ejs-button>
</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
<div class="sub-title"> Click the button to open/close the Sidebar.</div>
<div style="padding:20px" class="center-align">
<ejs-button ejs-button id="toggle"  class="e-btn e-info" v-
on:click="toggleClick" >Toggle Sidebar</ejs-button>
</div>
</div>
</div>
</div>
</template>
<script setup>
import { SidebarComponent as EjsSidebar } from '@syncfusion/ej2-vue-
navigations';
import { ButtonComponent as EjsButton } from '@syncfusion/ej2-vue-buttons';
import { ref } from "vue";
const type = 'Push';
const width = '280px';
const enableGestures = false;
const sidebar = ref(null);
const closeClick = () => {
sidebar.value.hide();
}
const toggleClick = () => {
sidebar.value.toggle();
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.center-align {
text-align: center;
padding: 20px;
}
.title {
text-align: center;
font-size: 20px;
padding: 15px;
}
.sub-title {
text-align: center;
font-size: 16px;
padding: 10px;
}
.center {
text-align: center;
display: none;
font-size: 13px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
font-weight: 400;
margin-top: 20px;
}
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
.close-btn:hover {
color: #fafafa;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class="wrapper">
<ejs-sidebar id="default-sidebar" ref="sidebar" :width="width" :type="type"
:enableGestures="enableGestures">
<div class="title"> Sidebar content</div>
<div class="sub-title">
Click the button to close the Sidebar.
</div>
<div class="center-align">
<ejs-button  id="close" v-on:click="closeClick"  class="e-btn close-
btn">Close Sidebar</ejs-button>
</div>
</ejs-sidebar>
<div>
<div class="title">Main content</div>
<div class="sub-title"> Click the button to open/close the Sidebar.</div>
<div style="padding:20px" class="center-align">
<ejs-button ejs-button id="toggle"  class="e-btn e-info" v-
on:click="toggleClick" >Toggle Sidebar</ejs-button>
</div>
</div>
</div>
</div>
</template>
<script>
import { SidebarComponent } from '@syncfusion/ej2-vue-navigations';
import { ButtonComponent } from '@syncfusion/ej2-vue-buttons';
//Component registration
export default {
name: "App",
components: {
"ejs-sidebar": SidebarComponent,
"ejs-button": ButtonComponent
},
data() {
return {
type: 'Push',
width: '280px',
enableGestures: false,
}
},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
methods: {
toggleClick: function () {
this.$refs.sidebar.toggle();
},
closeClick: function () {
this.$refs.sidebar.hide();;
},
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.center-align {
text-align: center;
padding: 20px;
}
.title {
text-align: center;
font-size: 20px;
padding: 15px;
}
.sub-title {
text-align: center;
font-size: 16px;
padding: 10px;
}
.center {
text-align: center;
display: none;
font-size: 13px;
font-weight: 400;
margin-top: 20px;
}
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
.close-btn:hover {
color: #fafafa;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Custom context in Vue Sidebar component

Sidebar has a flexible option to make it initialize, target to any HTML element alongside of the main content of a web page.

By default, Sidebar initializes target to the body element. Using the target property, set target element to initialize the Sidebar inside any HTML element apart from the body element.

If required , zIndex can be set when sidebar act as overlay type.

APP.VUE

```
<template>
<div id="app">
   <ejs-sidebar id="default-sidebar" ref="sidebar" :type="type"
:target="target">
        <div class="title"> Sidebar content</div>
        <div class="sub-title">
           Click the button to close the Sidebar.
        </div>
        <div class="center-align">
          <button ejs-button id="close" v-on:click="closeClick" class="e-btn
close-btn">Close Sidebar</button>
        </div>
    </ejs-sidebar>
        <div id="head">
          <ejs-button id="toggle"  ref="togglebtn" cssClass="e-flat"
iconCss="e-icons burg-icon" isToggle="true" v-
on:click.native="btnClick">Open</ejs-button>
        </div>
        <div>
        <div id="maincontent" class="content">
            <div>
                <div class="title">Main content</div>
```

```
                <div class="center-align">
                   content goes here.
                </div>
            </div>
        </div>
        </div>
</div>
<!--end of main content declaration -->
</template>
<script>
import Vue from 'vue';
import { SidebarPlugin } from '@syncfusion/ej2-vue-navigations';
import { ButtonPlugin} from '@syncfusion/ej2-vue-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
Vue.use(SidebarPlugin);
Vue.use(ButtonPlugin);
export default {
    data () {
        return {
         type :'Push',
         target : '.content'
        }
    },
    methods: {
        btnClick: function(){
        if(this.$refs.togglebtn.$el.classList.contains('e-active')){
            this.$refs.togglebtn.Content = 'Open';
            this.$refs.sidebar.hide();
        }
        else{
            this.$refs.togglebtn.Content = 'Close';
            this.$refs.sidebar.show();
        }
    },
        closeClick: function() {
         this.$refs.sidebar.hide();
         this.$refs.togglebtn.$el.classList.remove('e-active');
         this.$refs.togglebtn.Content = 'Open';
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.center-align {
    text-align: center;
    padding: 20px;
}
.burg-icon:before {
    content: '\e10d';
    font-size: 16px;
}
.title {
    text-align: center;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    font-size: 20px;
    padding: 15px;
}
#head {
    border: 1px solid #424242;
    border-bottom-color: transparent;
    background: #00897B;
}
#toggle,#container .e-btn.e-info:hover,#toggle:focus
{/*csslintallow:adjoining-classes*/
    background: #00695C;
    box-shadow: none;
    border-radius: 0;
    height: 39px;
    width: 100px;
}
#close,#close:hover,#close:active,#close:focus{ /* csslint allow: adjoining-
classes*/
    background: #fafafa;
    color: black
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
#default-sidebar {
    background-color: #26A69A;
    color: #ffffff;
}
.content{
    height:305px;
    border: 1px solid grey;
}
.close-btn:hover {
    color: #fafafa;
}
#toggle {
    color: white;
}
@font-face {
    font-family: 'e-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAkAAAAg4aGVhZBLt+DYAAADQAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAAYAAAAA
wbG9jYQukCgIAAAIkAAAAGm1heHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAzoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAAAAADAABAAAAAQAAlbrm7l8
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAD8wPzAAAACAACAAAAAAAAAAAEAAAAMAQIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAAgAAAAMAAAAUAAMAAQA
AABQABABeAAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAgABwAKAAkABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAAAAAIAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAA+oD6gALAAATCQEXCQEHCQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAD6gOkAAMABwALAAA3IRUhESEVIREhFSE
VA9b8KgPW/CoD1vwq6I0B64wB640AAAEAAAAA+oD4QCaAAABMx8aHQEPDjEPAh8bIT8bNS8SPxs
```

```
CAA0aGhgMDAsLCwoKCgkJCQgHBwYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHBoZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAwIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICAkJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBwgICAkKCgs
LDAwNDQ4ODxAQEBEWFxYWFhYVFRQUExIRERAOFxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEhERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTExUUFhUWFxYWFxEQEBAPDg4
NDQwMCwsKCgkICAgHBgYFBQQEBQQAAAAAAwAAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8fFz8XLxcPBgI+BQQDAwMCAT8EBAM
DAwIBAQIDAwMEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAwMDAgEBAgMDAwQE0AUEAwMDAgEBAgMDAwQ
FfAUEAwMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEB0AECAwMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAwMDAgEBAgMDAwQFaAQEAwQCAgElERMVFhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxYVExEQDgsJCAUDAQEDBQcKCw4QERMVFhcZGxsdHh4fICAhIiIhICAfHh4dHBoZFxYVExE
QDgsKBwUDAQEDBQcKCw4AAAIAAAAAA9MD6QALAE8AAAEOAQcuASc+ATceAQEHBgcnJgYPAQYWHwE
GFBcHDgEfAR4BPwEWHwEeATsBMjY/ATY3FxY2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGAps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAwdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYMhhSBhIIrgg
FAzIfE4QJDAwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOECQwMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4zA+v+Ff4VHwHMAc0AAAAAAQAAAAADAQPqAAUAAAEXCQEHAQLlHf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAAD8wPzAEAAgQEBAAAlDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUfHTsBPx09AS8dKwEPHQL1DQ0ODg4PDw8QEBAQERERERUUFBQTExITERERERBAPDw0ODAw
LCwkJCAcGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAgQDBQUGBgcHCQgJCv3QDQ0ODg4PDw8QEBA
QERERERUUFBQTExITERERERBAPDw0ODAwLCwkJCAcGBgQEAgL8fgIDBQUHCAkKCwwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcYFhYWFRQUExIREQ8PDg0MCwoJCAcFBQMCAgMFBQcICQoLDA0ODw8
RERITFBQVFhYWGBcYGRkZGhoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETEhMTFBQUFRERERERQEBAQDw8PDg4ODQ31ERE
RERAQEBAPDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETEhM
TFBQUFRoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBU
WFhYYFxgZGRkaGhkZGRgXGBYWFhUUFBMSEREPDw4NDAsKCQgHBQUDAgIDBQUHCAkKCwwNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgwMDAsKCggIBwU
FAwIBAQIDBQUHCAgKCgsMDAwODf00DQ4MDAwLCgoICAcFBQMCAQECAwUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgsLDA0NDv00Dg0NDAsLCgoICAcFBQMCAQECAwUFBwgICgoLCwwNDQ4CzA4NDQwLCwo
KCAgHBQUDAgAAABIA3gABAAAAAAAAAAEAAAABAAAAAAABAA0AAQABAAAAAAACAAcADgABAAAAAAA
DAA0AFQABAAAAAAAEAA0AIgABAAAAAAAFAAsALwABAAAAAAAGAA0AOgABAAAAAAAKACwARwABAAA
AAAALABIAcwADAAEECQAAAAIAhQADAAEECQABABoAhwADAAEECQACAA4AoQADAAEECQADABoArwA
DAAEECQAEABoAyQADAAEECQAFABYA4wADAAEECQAGABoA+QADAAEECQAKAFgBEwADAAEECQALACQ
BayBlLWljb25zLW1ldHJvUmVndWxhcmUtaWNvbnMtbWV0cm9lLWljb25zLW1ldHJvVmVyc2lvbiA
xLjBlLWljb25zLW1ldHJvRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwBlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQByAHMAaQBvAG4AIAAxAC4AMABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAB
NAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwBAgEDAQQBBQEGAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMHbWVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHQMY2hldnJvbi1sZWZ0CE1UX0NsZWFyDE1UX0p1bmttYWlscwR
zdG9wAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/position-cs1" %}

Variations in Vue Sidebar component

The Sidebar component's expand behaviour can be modified based on the purpose of use.

Expanding types of sidebar

The Sidebar can be set to initialize based on four different types that are consistent with the main component as explained below. When dataBind is invoked, this applies the pending property changes immediately to the component.

| Item | Description |

|-----|-----|

| Over | Sidebar floats over the main content area.|

| Push | Sidebar pushes the main content area to appear side-by-side, and shrinks the main content within the screen width.|

| Slide |Sidebar translates the x and y positions of main content area based on the Sidebar width. The main content area will not be adjusted within the screen width. |

| Auto | Sidebar with Over type in mobile resolution, and Push type in other higher resolutions. |

Auto is the default expand mode.

In the following sample, the types of Sidebar are demonstrated.

APP.VUE

```
<template>
<div id="app">
<ejs-sidebar id="default-sidebar" ref="sidebar" :type="type"
:target="target">
    <div class="title"> Sidebar content</div>
    <div class="sub-title">
        Click the button to close the Sidebar.
    </div>
    <div class="center-align">
        <button ejs-button id="close" v-on:click="closeClick" class="e-btn
close-btn">Close Sidebar</button>
    </div>
</ejs-sidebar>
<div id="head">
    <ejs-button id="toggle" ref="togglebtn"  cssClass="e-flat" iconCss="e-
icons burg-icon" isToggle="true" v-on:click.native="btnClick" >Open</ejs-
button>
</div>
<div>
<div id="maincontent" class="content">
    <div>
        <div class="title">Main content</div>
        <div class="rows">
            <div class="row">
                <ejs-radiobutton id='push' label="Push" name="type"
:change="changeHandler" ></ejs-radiobutton>
            </div>
            <div class="row">
                <ejs-radiobutton id='slide'  label="Slide" name="type"
:change="changeHandler"></ejs-radiobutton>
```

```
            </div>
            <div class="row">
                <ejs-radiobutton id='over'  label="Over" name="type"
:change="changeHandler"></ejs-radiobutton>
            </div>
            <div class="row">
                <ejs-radiobutton id='auto'  label="Auto" name="type"
checked="true" :change="changeHandler"></ejs-radiobutton>
            </div>
        </div>
    </div>
</div>
</div>
</div>
<!--end of main content declaration -->
</template>
<script>
import Vue from 'vue';
import { SidebarPlugin } from '@syncfusion/ej2-vue-navigations';
import { ButtonPlugin,RadioButtonPlugin} from '@syncfusion/ej2-vue-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
Vue.use(SidebarPlugin);
Vue.use(ButtonPlugin);
Vue.use(RadioButtonPlugin);
export default {
    data () {
        return {
         type :'Push',
         target : '.content'
        }
    },
    methods: {
        btnClick: function(){
        if(this.$refs.togglebtn.$el.classList.contains('e-active')){
            this.$refs.togglebtn.Content = 'Open';
            this.$refs.sidebar.hide();
        }
        else{
            this.$refs.togglebtn.Content = 'Close';
            this.$refs.sidebar.show();
        }
    },
        closeClick: function() {
         this.$refs.sidebar.hide();
         this.$refs.togglebtn.$el.classList.remove('e-active');
         this.$refs.togglebtn.Content = 'Open';
        },
        changeHandler:  function(args) {
        if(args.event.target.id == 'over') {
            this.type = 'Over';
            this.dataBind();
        } else if (args.event.target.id == 'push') {
             this.type = 'Push';
             this.dataBind();
        } else if (args.event.target.id == 'slide') {
             this.type = 'Slide';
             this.dataBind();
```

```
        } else {
            this.type = 'Auto';
            this.dataBind();
        }
    }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.rows{
    margin:auto;
    text-align:center;
}
.row{
    padding:10px;
    margin:auto;
}
.header {
    width:100%;
    height: 40px;
    font-size:20px;
    line-height: 40px;
    font-weight: 500;
    background: #eee;
    display: inline-block;
}
.center-align {
    text-align: center;
    padding: 20px;
}
.burg-icon:before{
    content: '\e10d';
    font-size:16px;
}
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
#head {
    border: 1px solid #424242;
    border-bottom-color:transparent;
    background:#00897B;
}
#toggle,#container .e-btn.e-info:hover,
#toggle:focus {/* csslint allow:   adjoining-classes*/
    background: #00695C;
    box-shadow: none;
    border-radius: 0;
    height: 39px;
    width: 100px;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
#close,#close:hover,#close:active,#close:focus{ /* csslint allow: adjoining-classes*/
    background: #fafafa;
    color:black
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
.radiobutton{
    display:inline-block;
    padding:10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
.content{
    height:305px;
    border: 1px solid grey;
}
  #default-sidebar {
    background-color: #26A69A;
    color: #ffffff;
}
.close-btn:hover {
    color: #fafafa;
}
#toggle {
    color:white;
}
@font-face {
    font-family: 'e-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAkAAAAg4aGVhZBLt+DYAAADQAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAAYAAAAA
wbG9jYQukCgIAAAIkAAAAGm1heHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAzoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAAAAADAABAAAAAQAAlbrm7l8
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAD8wPzAAAACAACAAAAAAAAAAAEAAAAMAQIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAAgAAAAMAAAAUAAMAAQA
AABQABABeAAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAgABwAKAAkABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAAAAAIAAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAAA+oD6gALAAATCQEXCQEHCQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAD6gOkAAMABwALAAA3IRUhESEVIREhFSE
VA9b8KgPW/CoD1vwq6I0B64wB640AAAEAAAAAA+oD4QCaAAABMx8aHQEPDjEPAh8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBwYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHBoZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAwIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICAkJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBwgICAkKCgs
LDAwNDQ4ODxAQEBEWFxYWFhYVFRQUExIRERAOFxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEhERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTExUUFhUWFxYWFxEQEBAPDg4
```

```
NDQwMCwsKCgkICAgHBgYFBQQEBQQAAAAAAwAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8fFz8XLxcPBgI+BQQDAwMCAT8EBAM
DAwIBAQIDAwMEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAwMDAgEBAgMDAwQE0AUEAwMDAgEBAgMDAwQ
FfAUEAwMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEB0AECAwMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAwMDAgEBAgMDAwQFaAQEAwQCAgElERMVFhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxYVExEQDgsJCAUDAQEDBQcKCw4QERMVFhcZGxsdHh4fICAhIiIhICAfHh4dHBoZFxYVExE
QDgsKBwUDAQEDBQcKCw4AAAIAAAAAA9MD6QALAE8AAAEOAQcuASc+ATceAQEHBgcnJgYPAQYWHwE
GFBcHDgEfAR4BPwEWHwEeATsBMjY/ATY3FxY2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGAps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAwdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYMhhSBhIIrgg
FAzIfE4QJDAwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOECQwMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4zA+v+Ff4VHwHMAc0AAAAAAQAAAAADAQPqAAUAAAEXCQEHAQLlHf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAD8wPzAEAAgQEBAAAlDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUfHTsBPx09AS8dKwEPHQL1DQ0ODg4PDw8QEBAQERERERUUFBQTExITEREREBAPDw0ODAw
LCwkJCAcGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAgQDBQUGBgcHCQgJCv3QDQ0ODg4PDw8QEBA
QERERERUUFBQTExITEREREBAPDw0ODAwLCwkJCAcGBgQEAgL8fgIDBQUHCAkKCwwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcYFhYWFRQUExIREQ8PDg0MCwoJCAcFBQMCAgMFBQcICQoLDA0ODw8
RERITFBQVFhYWGBcYGRkZGhoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBAREREREREQEBAQDw8PDg4ODQ31ERE
RERAQEBAPDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBAREREREREhM
TFBQUFRoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBU
WFhYYFxgZGRkaGhkZGRgXGBYWFhUUFBMSEREPDw4NDAsKCQgHBQUDAgIDBQUHCAkKCwwNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgwMDAsKCggIBwU
FAwIBAQIDBQUHCAgKCgsMDAwODf00DQ4MDAwLCgoICAcFBQMCAQECAwUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgsLDA0NDv00Dg0NDAsLCgoICAcFBQMCAQECAwUFBwgICgoLCwwNDQ4CzA4NDQwLCwo
KCAgHBQUDAgAAABIA3gABAAAAAAAAAAAEAAAABAAAAAAAABAA0AAQABAAAAAAACAAcADgABAAAAAAA
DAA0AFQABAAAAAAAAEAA0AIgABAAAAAAAAFAAsALwABAAAAAAAGAA0AOgABAAAAAAAKACwARwABAAA
AAAALABIAcwADAAEECQAAAAIAhQADAAEECQABABoAhwADAAEECQACAA4AoQADAAEECQADABoArwA
DAAEECQAEABoAyQADAAEECQAFABYA4wADAAEECQAGABoA+QADAAEECQAKAFgBEwADAAEECQALACQ
BayBlLWljb25zLW1ldHJvUmVndWxhcmUtaWNvbnMtbWV0cm9lLWljb25zLW1ldHJvVmVyc2lvbiA
xLjBlLWljb25zLW1ldHJvRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwBlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQByAHMAaQBvAG4AIAAxAC4AMABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAB
NAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwBAgEDAQQBBQEGAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMHbWVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHQMY2hldnJvbi1sZWZ0CE1UX0NsZWFyDE1UX0p1bmttYWlscwR
zdG9wAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/position-cs3" %}

See Also

- How to add sidebar with custom animation
- How to add multiple sidebar

Copyright © 2001 -2024 Syncfusion Inc.

Auto close in Vue Sidebar component

The Sidebar often behaves differently on mobile display and differently on desktop display. It has an effective feature that offers to set it in opened or closed state depending on the specified resolution. This is achieved through mediaQuery property that allows you to set the Sidebar in an expanded state or collapsed state only in user-defined resolution.

APP.VUE

```
<template>
    <div id="app">
        <ejs-sidebar id="default-sidebar" :width="width"
:mediaQuery="mediaQuery">
            <div class="title"> Sidebar content</div>
        </ejs-sidebar>
        <div>
            <div class="title">Main content</div>
            <div class="sub-title">
                * Sidebar will collapse and expand in based on screen
resolution automatically
            </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SidebarPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(SidebarPlugin);
export default {
    data() {
        return {
            width: '280px',
            mediaQuery: window.matchMedia('(min-width: 600px)'),
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 12px;
    padding: 10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    margin-top: 20px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/getting-started-cs1" %}

Open close

- In the following sample,the Sidebar will be in an expanded state only in resolution below 400px.

APP.VUE

```
<template>
    <div id="app">
        <ejs-sidebar id="default-sidebar" :width="width"
:mediaQuery="mediaQuery">
            <div class="title"> Sidebar content</div>
        </ejs-sidebar>
        <div>
            <div class="title">Main content</div>
            <div class="sub-title">
                * Sidebar will collapse and expand in based on screen
resolution automatically
            </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SidebarPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(SidebarPlugin);
export default {
    data() {
        return {
            width: '280px',
            mediaQuery: window.matchMedia('(max-width: 400px)'),
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    text-align: center;
    font-size: 12px;
    padding: 10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/getting-started-cs2" %}

Docking sidebar in Vue Sidebar component

Dock state of the Sidebar reserves some space on the page that always remains in a visible state when the Sidebar is collapsed. It is used to show the short term of a content like icons alone instead of lengthy text. To achieve this , set enableDock as true along with required dockSize.

APP.VUE

```
<template>
    <div id="app">
        <div id="wrapper">
            <ejs-sidebar id="dockSidebar" :enableDock='enableDock'
:width='width' :dockSize='dockSize' ref="dockSidebar"
                :position="position">
                <div class="dock">
                    <ul>
                        <li class="sidebar-item" id="toggle" v-
on:click="toggleClick">
                            <span class="e-icons expand"></span>
                            <span class="e-text" title="menu">Menu</span>
                        </li>
                        <li class="sidebar-item">
                            <span class="e-icons home"></span>
                            <span class="e-text" title="home">Home</span>
                        </li>
                        <li class="sidebar-item">
                            <span class="e-icons profile"></span>
                            <span class="e-text"
title="profile">Profile</span>
                        </li>
                        <li class="sidebar-item">
                            <span class="e-icons info"></span>
                            <span class="e-text" title="info">Info</span>
                        </li>
                        <li class="sidebar-item">
                            <span class="e-icons settings"></span>
                            <span class="e-text"
title="settings">Settings</span>
```

```
                        </li>
                    </ul>
                </div>
            </ejs-sidebar>
            <div id="main-content container-fluid col-md-12 ">
                <div class="title">Main content</div>
                <div class="sub-title"> Click the expand icon to open and
collapse icons to close the Sidebar</div>
            </div>
        </div>
    </div>
    <!--end of main content declaration -->
</template>
<script>
import Vue from "vue";
import { SidebarPlugin } from '@syncfusion/ej2-vue-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
Vue.use(SidebarPlugin);
export default {
    data() {
        return {
            enableDock: true,
            dockSize: '72px',
            width: '220px',
            position: 'Left'
        }
    },
    methods: {
        toggleClick: function () {
            this.$refs.dockSidebar.toggle();
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
#wrapper .column {
    display: inline-block;
    padding: 10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    font-weight: 400;
    margin-top: 20px;
}
.sb-content-tab .center {
    display: block;
}
/* end of content area styles */
/* Sidebar styles */
#dockSidebar.e-sidebar.e-right.e-close {
    visibility: visible;
    transform: translateX(0%);
}
#dockSidebar .e-icons::before {
    font-size: 25px;
}
/* dockbar icon Style */
#dockSidebar .home::before {
    content: '\e102';
}
#dockSidebar .profile::before {
    content: '\e10c'
}
#dockSidebar .info::before {
    content: '\e11b';
}
#dockSidebar .settings::before {
    content: '\e10b';
}
.e-sidebar .expand::before {
    content: '\e10f';
}
.e-sidebar.e-open .expand::before {
    /* csslint allow: adjoining-classes*/
    content: '\e10e';
}
/* end of dockbar icon Style */
#dockSidebar.e-close .sidebar-item {
    padding: 5px 20px;
}
#dockSidebar.e-dock.e-close span.e-text {
    display: none;
}
#dockSidebar.e-dock.e-open span.e-text {
    display: inline-block;
}
#dockSidebar li {
    list-style-type: none;
    cursor: pointer;
}
#dockSidebar ul {
    padding: 0px;
}
#dockSidebar.e-sidebar ul li:hover span {
    color: white
}
#dockSidebar.e-sidebar.e-open .e-text {
    overflow: hidden;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    text-overflow: ellipsis;
    line-height: 23px;
    font-size: 15px;
}
#dockSidebar.e-sidebar.e-open .e-icons {
    margin-right: 16px;
}
#dockSidebar.e-sidebar span.e-icons {
    color: #c0c2c5;
    line-height: 2
}
#dockSidebar.e-sidebar .e-open .e-icons {
    margin-right: 16px;
}
#dockSidebar.e-sidebar .e-open .e-text {
    overflow: hidden;
    text-overflow: ellipsis;
    line-height: 23px;
    font-size: 15px;
}
.sidebar-item {
    text-align: center;
    border-bottom: 1px #e5e5e58a solid;
}
#dockSidebar.e-sidebar.e-open .sidebar-item {
    text-align: left;
    padding-left: 15px;
    color: #c0c2c5;
}
#dockSidebar.e-sidebar {
    background: #2d323e;
    overflow: hidden;
}
@font-face {
    font-family: 'e-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAkAAAAg4aGVhZBLt+DYAAADQAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAAYAAAAA
wbG9jYQukCgIAAAIkAAAAGm1heHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAzoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAAAAAADAABAAAAAQAAlbrm718
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAD8wPzAAAACAACAAAAAAAAAAAEAAAAMAQIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAAgAAAAMAAAAUAAMAAQA
AABQABABeAAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAgABwAKAAkABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAAAAAIAAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAAA+oD6gALAAATCQEXCQEHCQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAD6gOkAAMABwALAAA3IRUhESEVIREhFSE
VA9b8KgPW/CoD1vwq6I0B64wB640AAAEAAAAAA+oD4QCaAAABMx8aHQEPDjEPAh8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBwYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHBoZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAwIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICAkJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBwgICAkKCgs
LDAwNDQ4ODxAQEBEWFxYWFhYVFRQUExIRERAOFxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEhERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTExUUFhUWFxYWFxEQEBAPDg4
NDQwMCwsKCgkICAgHBgYFBQQEBQQAAAAAAwAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8fFz8XLxcPBgI+BQQDAwMCAT8EBAM
```

```
DAwIBAQIDAwMEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAwMDAgEBAgMDAwQE0AUEAwMDAgEBAgMDAwQ
FfAUEAwMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEB0AECAwMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAwMDAgEBAgMDAwQFaAQEAwQCAgElERMVFhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxYVExEQDgsJCAUDAQEDBQcKCw4QERMVFhcZGxsdHh4fICAhIiIhICAfHh4dHBoZFxYVExE
QDgsKBwUDAQEDBQcKCw4AAAIAAAAAA9MD6QALAE8AAAEOAQcuASc+ATceAQEHBgcnJgYPAQYWHwE
GFBcHDgEfAR4BPwEWHwEeATsBMjY/ATY3FxY2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGAps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAwdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYMhhSBhIIrgg
FAzIfE4QJDAwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOECQwMAAEAAAAAAwED6gAFAAAAJAic
JAQEbAef+FhoBzf4zA+v+Ff4VHwHMAc0AAAAAAQAAAAADAQPqAAUAAAEXCQEHAQLlHf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAD8wPzAEAAgQEBAAAlDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUfHTsBPx09AS8dKwEPHQL1DQ0ODg4PDw8QEBAQERERERUUFBQTExITERERERBAPDw0ODAw
LCwkJCAcGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAgQDBQUGBgcHCQgJCv3QDQ0ODg4PDw8QEBA
QERERERUUFBQTExITERERERBAPDw0ODAwLCwkJCAcGBgQEAgL8fgIDBQUHCAkKCwwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcYFhYWFRQUExIREQ8PDg0MCwoJCAcFBQMCAgMFBQcICQoLDA0ODw8
RERITFBQVFhYWGBcYGRkZGhoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBAREReTEhMTFBQUFRERERERQEBAQDw8PDg4ODQ31ERE
RERAQEBAPDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBAREReTEhM
TFBQUFRoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBU
WFhYYFxgZGRkaGhkZGRgXGBYWFhUUFBMSEREPDw4NDAsKCQgHBQUDAgIDBQUHCAkKCwwNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgwMDAsKCggIBwU
FAwIBAQIDBQUHCAgKCgsMDAwODf00DQ4MDAwLCgoICAcFBQMCAQECAwUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgsLDA0NDv00Dg0NDAsLCgoICAcFBQMCAQECAwUFBwgICgoLCwwNDQ4CzA4NDQwLCwo
KCAgHBQUDAgAAABIA3gABAAAAAAAAAAEAAAABAAAAAAAABAA0AAQABAAAAAAACAAcADgABAAAAAAA
DAA0AFQABAAAAAAAAEAA0AIgABAAAAAAAAFAAsALwABAAAAAAAGAA0AOgABAAAAAAAKACwARwABAAA
AAAALABIAcwADAAEECQAAAAIAhQADAAEECQABABoAhwADAAEECQACAA4AoQADAAEECQADABoArwA
DAAEECQAEABoAyQADAAEECQAFABYA4wADAAEECQAGABoA+QADAAEECQAKAFgBEwADAAEECQALACQ
BayBlLWljb25zLW1ldHJvUmVndWxhcmUtaWNvbnMtbWV0cm9lLWljb25zLW1ldHJvVmVyc2lvbiA
xLjBlLWljb25zLW1ldHJvRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwBlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQByAHMAaQBvAG4AIAAxAC4AMABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAB
NAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwBAgEDAQQBBQEGAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMHbWVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHQMY2hldnJvbi1sZWZ0CE1UX0NsZWFyDE1UX0p1bmttYWlscwR
zdG9wAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/getting-started-cs3" %}

See Also

- How to add sidebar navigation

Style in Vue Sidebar component

The following content provides the exact CSS structure that can be used to modify the component's appearance based on the user's preference.

Customizing the sidebar

Use the below CSS to customize the sidebar root element.

```
.e-sidebar {
background: #898b2b
}
```

Customizing the sidebar based on the positions

Use the below CSS to customize the left positioned sidebar.

```
.e-sidebar.e-left {
border-right: 2px solid red;
}
```

Use the below CSS to customize the right positioned sidebar.

```
.e-sidebar.e-right {
border-left: 2px solid red;
}
```

Customizing the sidebar based on the active state

Use the below CSS to customize the open state of the left positioned sidebar.

```
.e-sidebar.e-left.e-open {
transition: transform 2.5s ease;
}
```

Use the below CSS to customize the open state of the right positioned sidebar.

```
.e-sidebar.e-right.e-open {
transition: transform 2.5s ease;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
```

Use the below CSS to customize the closed state of the left positioned sidebar.

```
.e-sidebar.e-left.e-transition.e-close {
transition: transform 2.5s ease, visibility 1200ms;
}
```

Use the below CSS to customize the closed state of the right positioned sidebar.

```
.e-sidebar.e-right.e-transition.e-close {
transition: transform 2.5s ease, visibility 1200ms;
}
```

Customizing the sidebar with dock state

When you enable the Dock support, the "e-dock" class will be added to the root element. Based on that class, you can also customize all the above stated customization. Use the following CSS to customize the sidebar element with a dock state.

```
.e-sidebar.e-dock {
background: #2d323e;
}
```

Customizing the different types of sidebar

Use the below CSS to customize the auto type sidebar.

```
.e-sidebar.e-left.e-auto {
background-color: pink;
}
```

Use the below CSS to customize the push type sidebar.

```
.e-sidebar.e-left.e-push {
background-color: beige;
```

```
}
```

Use the below CSS to customize the over type sidebar.

```
.e-sidebar.e-left.e-over {
background-color: aqua;
}
```

Use the below CSS to customize the slide type sidebar.

```
.e-sidebar.e-left.e-slide {
background-color: green;
}
```

Customizing the backdrop of the sidebar

Use the below CSS to customize the backdrop of the sidebar.

```
.e-sidebar-overlay {
background-color: aqua;
}
```

Customizing the sidebar in the RTL direction

When you enable the RTL (right to left direction) support, the "e-rtl" class will be added to the root element. Based on that class, you can also customize all the above stated customization. Use the following CSS to customize the sidebar element in the RTL (right to left direction) mode.

```
.e-sidebar.e-left.e-rtl {
background-color: antiquewhite;
}
```

Prevent the animation transition for the Sidebar component

Use the below CSS to prevent the animation transition for the Sidebar component.

```
.e-sidebar-context .e-content-animation {
transition: none !important;
```

Copyright © 2001 -2024 Syncfusion Inc.

transform: none !important;

}

`

Accessibility in Vue Sidebar component

The Sidebar component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Sidebar component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

Copyright © 2001 -2024 Syncfusion Inc.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Sidebar component followed the WAI-ARIA patterns to meet accessibility standards. By default, the Sidebar utilizes the complementary role, with the option to modify the ARIA role based on provided attributes to the root element, depending on the specific use case.

If there are multiple complementary landmark roles or aside elements in a document, it is important to provide a label for each landmark using the `aria-label` attribute. Alternatively, if the aside has a descriptive title, it can be referenced using the `aria-labelledby` attribute. This label will help assistive technology users quickly understand the purpose of each landmark.

For optimal accessibility, it is recommended to incorporate a trigger component that is navigable via a keyboard, such as a button. If this is not feasible, inclusion of the `tabIndex` attribute becomes necessary. Furthermore, explicit handling of the `aria-expanded` attribute is required for the trigger element.

Keyboard interaction

The Sidebar component does not have any inbuilt keyboard interaction support. However, you can utilize the keyboard interactions of focusable elements within the Sidebar component.

Ensuring accessibility

The Sidebar component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Sidebar component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Sidebar component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/side-bar.html" %}

See also

- Accessibility in Syncfusion Vue components

How To

Sidebar with listview in Vue Sidebar component

Any HTML element can be placed in the Sidebar content area. Sidebar supports all types of HTML structures like `TreeView`, `ListView`, etc.

In the following example, the Sidebar is rendered with ListView component in its content area.

APP.VUE

```
<template>
<div id="app">
        <ejs-sidebar id="default-sidebar" ref="sidebar" :width="width"
:type="type">
    <div class="title1">Menu</div>
            <div class="closebtn">
                <button  ejs-button id="close" class="e-btn close-btn" v-
on:click="closeClick">
                    <span id="innerclose" class="e-icons close-icon"></span>
                </button>
```

```
            </div>
            <div id="listcontainer">
          <ejs-listview id='list' :dataSource="dataList" ></ejs-listview>
          </div>
              <div class="sub-title">
                     * ListView component is rendered in the sidebar
container.
                </div>
</ejs-sidebar>
<div>
    <div>
        <div class="title2">Main content</div>
        <div class="sub-title"> Click the button to open/close the Sidebar.
            <div style="padding:20px" class="center-align">
                <button ejs-button id="toggle" class="e-btn e-info" v-
on:click="toggleClick">Toggle Sidebar</button>
            </div>
        </div>
    </div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { SidebarPlugin } from '@syncfusion/ej2-vue-navigations';
import { ListViewPlugin, SelectEventArgs } from '@syncfusion/ej2-vue-lists';
import { enableRipple } from '@syncfusion/ej2-base';
Vue.use(SidebarPlugin);
Vue.use(ListViewPlugin);
Vue.use(SidebarPlugin);
Vue.use(ListViewPlugin);
export default {
    data () {
           return {
          dataList: [
        { text: 'Home', id: 'list-02' },
        { text: 'Offers', id: 'list-03' },
        { text: 'Support', id: 'list-04' },
        { text: 'Logout', id: 'list-06' }
    ],
    type: 'Over',
    width: '100%'
     }
     },
     methods: {
        toggleClick :function() {
          this.$refs.sidebar.toggle();
        },
        closeClick: function() {
         this.$refs.sidebar.hide();
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
```

```
.center-align {
    text-align: center;
    padding: 20px;
}
.close-btn,.e-listview .e-list-item, #default-sidebar {
    background-color: rgb(20, 118, 210);
    color: #ffffff;
}
.close-btn {
    box-shadow: none;
}
.closebtn {
    top: 15px;
    line-height: 36px;
    height: 42px;
    color: black;
    position: absolute;
    right: 10px;
}
#listcontainer {
    width: 100%;
}
#list {
    margin: 0 auto;
    width: 30%;
}
.close-icon::before {
    content: "\e109";
}
.e-listview .e-list-item {
    text-align: center;
    font-size: 14px;
    padding: 0;
}
.e-btn.close-btn :hover { /* csslint allow: adjoining-classes*/
    box-shadow: none;
    background: transparent;
}
.e-icons.close-icon { /* csslint allow: adjoining-classes*/
    line-height: 2.2;
}
.title1 {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.title2 {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 12px;
    padding: 10px;
}
.center {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
.close-btn:hover {
    color: #fafafa;
}
.content-section { /* csslint allow: adjoining-classes */
    padding: 30px 10px 10px 20px;
}
@font-face {
    font-family: 'e-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAkAAAAg4aGVhZBLt+DYAAADQAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAAYAAAAA
wbG9jYQukCgIAAAIkAAAAGm1heHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAzoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAAAAAADAABAAAAAQAAlbrm718
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAD8wPzAAAACAACAAAAAAAAAAAEAAAAMAQIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAAAgAAAAMAAAAUAAMAAQA
AABQABABeAAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAgABwAKAAkABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAAAAAIAAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAA+oD6gALAAATCQEXCQEHCQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAD6gOkAAMABwALAAA3IRUhESEVIREhFSE
VA9b8KgPW/CoD1vwq6I0B64wB640AAAEAAAAAA+oD4QCaAAABMx8aHQEPDjEPAh8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBwYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHBoZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAwIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICAkJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBwgICAkKCgs
LDAwNDQ4ODxAQEBEWFxYWFhYVFRQUExIRERAOFxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEhERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTExUUFhUWFxYWFxEQEBAPDg4
NDQwMCwsKCgkICAgHBgYFBQQEBQQAAAAAAwAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8fFz8XLxcPBgI+BQQDAwMCAT8EBAM
DAwIBAQIDAwMEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAwMDAgEBAgMDAwQE0AUEAwMDAgEBAgMDAwQ
FfAUEAwMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEB0AECAwMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAwMDAgEBAgMDAwQFaAQEAwQCAgElERMVFhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxYVExEQDgsJCAUDAQEDBQcKCw4QERMVFhcZGxsdHh4fICAhIiIhICAfHh4dHBoZFxYVExE
QDgsKBwUDAQEDBQcKCw4AAAIAAAAAA9MD6QALAE8AAAEOAQcuASc+ATceAQEHBgcnJgYPAQYWHwE
GFBcHDgEfAR4BPwEWHwEeATsBMjY/ATY3FxY2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGAps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAwdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYMhhSBhIIrgg
FAzIfE4QJDAwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOECQwMAAEAAAAAAwED6gAFAAAAJAic
JAQEbAef+FhoBzf4zA+v+Ff4VHwHMAc0AAAAAAQAAAAADAQPqAAUAAAEXCQEHAQLlHf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAD8wPzAEAAgQEBAAAlDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUfHTsBPx09AS8dKwEPHQL1DQ0ODg4PDw8QEBAQERERERUUFBQTExITEREREBAPDw0ODAw
LCwkJCAcGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAgQDBQUGBgcHCQgJCv3QDQ0ODg4PDw8QEBA
QERERERUUFBQTExITEREREBAPDw0ODAwLCwkJCAcGBgQEAgL8fgIDBQUHCAkKCwwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcYFhYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETEhMTFBQUFRERERERAQEBAPDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETEhM
```

```
TFBQUFRoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBU
WFhYYFxgZGRkaGhkZGRgXGBYWFhUUFBMSEREPDw4NDAsKCQgHBQUDAgIDBQUHCAkKCwwNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgwMDAsKCggIBwU
FAwIBAQIDBQUHCAgKCgsMDAwODf00DQ4MDAwLCgoICAcFBQMCAQECAwUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgsLDA0NDv00Dg0NDAsLCgoICAcFBQMCAQECAwUFBwgICgoLCwwNDQ4CzA4NDQwLCwo
KCAgHBQUDAgAAABIA3gABAAAAAAAAAAEAAAABAAAAAAAABAA0AAQABAAAAAAACAAcADgABAAAAAAA
DAA0AFQABAAAAAAAEAA0AIgABAAAAAAAFAAsALwABAAAAAAAGAA0AOgABAAAAAAAKACwARwABAAA
AAAALABIAcwADAAEECQAAAAIAhQADAAEECQABABoAhwADAAEECQACAA4AoQADAAEECQADABoArwA
DAAEECQAEABoAyQADAAEECQAFABYA4wADAAEECQAGABoA+QADAAEECQAKAFgBEwADAAEECQALACQ
BayBlLWljb25zLW1ldHJvUmVndWxhcmUtaWNvbnMtbWV0cm9lLWljb25zLW1ldHJvVmVyc2lvbiA
xLjBlLWljb25zLW1ldHJvRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwBlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQByAHMAaQBvAG4AIAAxAC4AMABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAB
NAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwBAgEDAQQBBQEGAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMHbWVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHQMY2hldnJvbi1sZWZ0CE1UX0NsZWFyDE1UX0p1bmttYWlscwR
zdG9wAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/listview-cs2" %}

Open close sidebar in Vue Sidebar component

Opening and closing the Sidebar can be achieved with built-in public methods.

- show(): Method to open the Sidebar.
- hide(): Method to close the Sidebar.
- toggle(): Method to toggle between open and close states of the Sidebar.

In the following sample, toggle method has been used to show or hide the Sidebar on button click.

APP.VUE

```
<template>
<div id="app">
        <ejs-sidebar id="default-sidebar" ref="sidebar" :open="open"
:close="close" :showBackdrop="showBackdrop">
    <div class="title"> Sidebar content</div>
    <div class="sub-title">
        Click the button to close the Sidebar.
    </div>
    <div class="center-align">
        <button ejs-button id="close" v-on:click="closeClick" class="e-btn
close-btn">Close Sidebar</button>
    </div>
</ejs-sidebar>
<div>
    <div>
        <div class="title">Main content</div>
        <div class="sub-title"> Click the button to Open the Sidebar.
```

```
            <div style="padding:20px" class="center-align">
                <button ejs-button id="open" class="e-btn e-info" v-
on:click="openClick">Open Sidebar</button>
            </div>
        </div>
        <div class="sub-title"> Click the button to open/close the Sidebar.
            <div style="padding:20px" class="center-align">
                <button ejs-button id="toggle" class="e-btn e-info" v-
on:click="toggleClick">Toggle Sidebar</button>
            </div>
        </div>
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { SidebarPlugin } from '@syncfusion/ej2-vue-navigations';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SidebarPlugin);
Vue.use(ButtonPlugin);
export default {
     data () {
          return {
            showBackdrop :false,
        };
     },
     methods: {
        openClick :function() {
          this.$refs.sidebar.show();
        },
        toggleClick :function() {
          this.$refs.sidebar.toggle();
        },
        closeClick: function() {
         this.$refs.sidebar.hide();
        },
        close:function() {
            console.log("sidebar closed");
        },
        open:function() {
            console.log("sidebar opened");
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.center-align {
    text-align: center;
    padding: 20px;
}
.title {
    text-align: center;
    font-size: 20px;
```

```
    padding: 15px;
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
}
.center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
}
#default-sidebar {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
.close-btn:hover {
    color: rgba(0, 0, 0, .87);
    background-color: #fafafa;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/howto-cs2" %}

Multiple sidebar in Vue Sidebar component

Two Sidebars can be initialized in a web page with same main content. Sidebars can be initialized on right side or left side of the main content using position property.

The HTML element with class name e-main-content will be considered as the main content and both the Sidebars will behave as side content to this main content area.

APP.VUE

```
<template>
<div id="app">
        <ejs-sidebar id="default" ref="leftsidebar" :width="leftWidth"
:type="leftType">
          <div class="title"> Left Sidebar content</div>
        </ejs-sidebar>
        <ejs-sidebar id="default1" ref="rightsidebar" :width="width"
:position="position"
           :type="type">
           <div class="title">Right Sidebar content</div>
        </ejs-sidebar>
    <div class="e-main-content">
           <p>Place your main content here.....</p>
        <div id="button-align">
            <button ejs-button id="toggle" class="e-btn e-info" v-
on:click="leftToggle">Toggle Sidebar1</button>
        </div>
        <div id="button-align">
          <button ejs-button id="toggle2" class="e-btn e-info" v-
on:click="rightToggle">Toggle Sidebar2</button>
        </div>
```

```
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { SidebarPlugin } from '@syncfusion/ej2-vue-navigations';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SidebarPlugin);
Vue.use(ButtonPlugin);
export default {
       data () {
          return {
          leftWidth: '250px'
          leftType: 'Push'
          width : '250px',
          position: 'Right',
          type: 'Push'
     }
     },
     methods: {
        leftToggle :function() {
          this.$refs.leftsidebar.toggle();
        },
        rightToggle: function() {
         this.$refs.rightsidebar.toggle();
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.e-main-content{
    text-align: center;
    height: 100vh;
    background: #f5f5f5;
    font-size: 16px;
    padding: 100px;
}
#button-align{
    padding: 15px;
}
.title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
}
#default,#default1 {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
    overflow: hidden;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/sidebar/howto-cs1" %}

Sidebar with treeview in Vue Sidebar component

The following example demonstrates how to render TreeView component inside the Sidebar with dock state and how to achieve expand and collapse the functionalities simultaneously in the sidebar and Treeview.

On collapse, the LI elements of TreeView show icons only to represent the short sign of the hidden text content. On expand, hidden text content will be set to be visible.

APP.VUE

```
<template>
    <div id="app">
       <div class="col-lg-8 control-section">
           <div class='main-header' id='header-section'>
                <ul class='header-list'>
                    <li class='float-left header-style icon-menu'
id='hamburger' v-on:click="toggleClick" ref="sidbarToggle"></li>
                    <li class='float-left header-style nav-
pane'><b>Navigation Pane</b></li>
                    <li class='header-style float-right support border-
left'><b>Support</b></li>
                </ul>
            </div>
            <!--Sidebar element declaration-->
            <ejs-sidebar id='sidebar-treeview' :enableDock='enableDock'
:width='width' :dockSize='dockSize' :mediaQuery='mediaQuery'
:target='target' ref="sidebarInstance" :close="close">
                <div class='main-menu'>
                    <div>
                    <ejs-treeview id='main-treeview' :fields='fields'
expandOn='Click'  ref="treeviewInstance" ></ejs-treeview>
                    </div>
                </div>
            </ejs-sidebar>
            <!--End of sidebar declaration-->
        </div>
         <div class='main-content' id='main-text'>
            <div class='sidebar-content'>
                <h2 class='sidebar-heading'> Responsive Sidebar With
Treeview</h2>
                <p class='paragraph-content'> This is a graphical aid for
visualising and categorising the site,
                in the style of an expandable and collapsable treeview
component. It auto-expands to display the node(s),
                 if any, corresponding to the currently viewed title,
highlighting that node(s) and its ancestors. Load-on-demand
                 when expanding nodes is available where supported (most
graphical browsers), falling back to a full-page reload.
                 MediaWiki-supported caching, aside from squid, has been
considered so that unnecessary re-downloads of content are
                  avoided where possible. The complete expanded/collapsed
state of the treeview persists across page views in most
                  situations</p>
                <p class='paragraph-content'>Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis
                    nostrud exercitation ullamco laboris nisi ut aliquip ex
ea commodo consequat. Duis aute irure
                    dolor in reprehenderit in voluptate velit esse cillum
dolore eu fugiat nulla pariatur.
                    Excepteur sint occaecat cupidatat non proident, sunt in
culpa qui officia deserunt mollit anim
                    id est laborum.</p>
                <div class='line'></div>
                <h2 class='sidebar-heading'>Lorem Ipsum Dolor</h2>
                <p class='paragraph-content'>Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do
                    eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis
                    nostrud exercitation ullamco laboris nisi ut aliquip ex
ea commodo consequat. Duis aute irure
                    dolor in reprehenderit in voluptate velit esse cillum
dolore eu fugiat nulla pariatur.</p>
                <div class='line'></div>
                <h2 class='sidebar-heading'> Lorem Ipsum Dolor</h2>
                <p class='paragraph-content'>Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do eiusmod
                    tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud
                    exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in
                    reprehenderit in voluptate velit esse cillum dolore eu
fugiat nulla pariatur. Excepteur sint
                    occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum.</p>
                 <div class='line'></div>
                <h2 class='sidebar-heading'> Lorem Ipsum Dolor</h2>
                <p class='paragraph-content'>Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do eiusmod
                    tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud
                    exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in
                    reprehenderit in voluptate velit esse cillum dolore eu
fugiat nulla pariatur. Excepteur sint
                    occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum.</p>
            </div>
        </div>
        <!--end of main content declaration -->
    </div>
</template>
<script>
import Vue from 'vue';
import { SidebarPlugin } from '@syncfusion/ej2-vue-navigations';
import { TreeViewPlugin } from '@syncfusion/ej2-vue-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SidebarPlugin);
Vue.use( TreeViewPlugin);
export default {
```

```
        data () {
            var data=[
             {
                  nodeId: '01', nodeText: 'Installation', iconCss: 'icon-
microchip icon',
                },
                {
                    nodeId: '02', nodeText: 'Deployment', iconCss: 'icon-
thumbs-up-alt icon',
                },
                {
                    nodeId: '03', nodeText: 'Quick Start', iconCss: 'icon-
docs icon',
                },
                {
                    nodeId: '04', nodeText: 'Components', iconCss: 'icon-th
icon',
                    nodeChild: [
                        { nodeId: '04-01', nodeText: 'Calendar', iconCss:
'icon-circle-thin icon' },
                        { nodeId: '04-02', nodeText: 'DatePicker', iconCss:
'icon-circle-thin icon' },
                        { nodeId: '04-03', nodeText: 'DateTimePicker',
iconCss: 'icon-circle-thin icon' },
                        { nodeId: '04-04', nodeText: 'DateRangePicker',
iconCss: 'icon-circle-thin icon' },
                        { nodeId: '04-05', nodeText: 'TimePicker', iconCss:
'icon-circle-thin icon' },
                        { nodeId: '04-06', nodeText: 'SideBar', iconCss:
'icon-circle-thin icon' }
                    ]
                },
                {
                    nodeId: '05', nodeText: 'API Reference', iconCss: 'icon-
code icon',
                    nodeChild: [
                        { nodeId: '05-01', nodeText: 'Calendar', iconCss:
'icon-circle-thin icon' },
                        { nodeId: '05-02', nodeText: 'DatePicker', iconCss:
'icon-circle-thin icon' },
                        { nodeId: '05-03', nodeText: 'DateTimePicker',
iconCss: 'icon-circle-thin icon' },
                        { nodeId: '05-04', nodeText: 'DateRangePicker',
iconCss: 'icon-circle-thin icon' },
                        { nodeId: '05-05', nodeText: 'TimePicker', iconCss:
'icon-circle-thin icon' },
                        { nodeId: '05-06', nodeText: 'SideBar', iconCss:
'icon-circle-thin icon' }
                    ]
                },
                {
                    nodeId: '06', nodeText: 'Browser Compatibility',
iconCss: 'icon-chrome icon'
                },
                {
                    nodeId: '07', nodeText: 'Upgrade Packages', iconCss:
'icon-up-hand icon'
```

```
                },
                {
                    nodeId: '08', nodeText: 'Release Notes', iconCss: 'icon-
bookmark-empty icon'
                },
                {
                    nodeId: '09', nodeText: 'FAQ', iconCss: 'icon-help-
circled icon'
                },
                {
                    nodeId: '10', nodeText: 'License', iconCss: 'icon-doc-
text icon'
                }
        ];
        return {
            fields: { child: 'nodeChild', dataSource: data, iconCss:
'iconCss' , id: 'nodeId', text: 'nodeText'},
            enableDock: true,
            dockSize : '44px',
            width : '290px',
            target: '.main-content',
            mediaQuery: '(min-width: 600px)',
        };
     },
     methods: {
        toggleClick: function() {
            if(this.$refs.sidebarInstance.ej2Instances.isOpen)
            {
                this.$refs.sidebarInstance.hide();
                this.$refs.treeviewInstance.ej2Instances.collapseAll();
            }
            else {
                this.$refs.sidebarInstance.show();
                this.$refs.treeviewInstance.ej2Instances.expandAll();
            }
        },
        close:function() {
           this.$refs.treeviewInstance.ej2Instances.collapseAll();
        },
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
/* header-section styles */
    #header-section.main-header {
        border-bottom: 1px solid #d2d6de;
        height: 55px;
        min-height: 55px;
        max-height: 55px;
        background: #1c86c8;
        color: #fff;
    }
    #header-section .header-style {
        line-height: 40px;
```

```
        height: 40px;
        padding: 8px;
        list-style: none;
        cursor: pointer;
        text-align: center;
        font-size: 18px;
    }
    #header-section .border-left {
        border-left: 1px solid #d2d6de;
        width: 10em;
    }
    #header-section .float-left {
        float: left;
        padding-right: 0px;
    }
    #header-section .icon-menu {
        width: 40px;
    }
    #header-section .header-list,
    #sidebar-treeview .e-treeview,
    #sidebar-treeview .e-treeview .e-ul {
        padding: 0px;
        margin: 0px;
    }
    /*end of header-section styles */
    /*main-menu-header  styles */
    #sidebar-treeview .main-menu .main-menu-header {
        color: #656a70;
    padding: 15px;
    font-size: 14px;
    width: 13em;
    margin: 0;
    }
    /*end of main-menu-header styles */
    /*text input styles */
    #sidebar-treeview .main-menu .search-icon {
        text-indent: 10px;
        height: 30px;
        width: 18em;
    }
    /*end of text input styles */
    /* table of content area styles */
    #sidebar-treeview .table-content {
        padding: 20px;
    height: 5em;
    }
    /* end of table ofcontent area styles */
    /* content area styles */
    #main-text.main-content {
        overflow: hidden;
    }
    #main-text .sidebar-content .line {
        width: 100%;
        height: 1px;
        border-bottom: 1px dashed #ddd;
        margin: 40px 0;
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    #main-text .sidebar-content {
        padding: 15px;
    }
    #main-text .sidebar-heading {
        color: #1c86c8;
        margin: 40px 0;
        padding: 2px;
    }
    #main-text .paragraph-content {
       font-family: 'Poppins', sans-serif;
       padding: 5px;
       font-weight: 300;
       color: grey;
    }
    /* end of content area styles */
    /* end of body and html styles */
    /* icon styles */
    @font-face {
        font-family: 'fontello';
        src:url('data:application/octet-
streambase64,AAEAAAAPAIAAAwBwR1NVQiCLJXoAAAD8AAAAVE9TLzI+JUkyAAABUAAAAFZjbW
Fw0almQAAAAagAAAIgY3Z0IAbV/vwAABfUAAAAIGZwZ22KkZBZAAAX9AAAC3BnYXNwAAAAEAAAF8
wAAAAIZ2x5Zk3OJrMAAAPIAAAPrGhlYWQTw6AfAAATdAAAADZoaGVhB2gDnAAAE6wAAAAkaG10eD
Hm//YAABPQAAAAOGxvY2EejhqYAAAUCAAAAB5tYXhwAfYMkAAAFCgAAAgbmFtZcydHiAAABRIAA
ACzXBvc3RuKDzPAAAXGAAAALRwcmVw5UErvAAAI2QAAACGAAEAAAAKADAAPgACREZMVAAObGF0bg
AaAAQAAAAAAAAAQAAAAQAAAAAAAAAAQAAAAFsaWdhAAgAAAABAAAAAQAEAAQAAAABAAgAAQAGAA
AAAQAAAAEDkAGQAAUAAAJ6ArwAAACMAnoCvAAAAeAAMQECAAACAAUDAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAFBmRWQAQOgB6BMDUv9qAFoDUgCaAAAAAQAAAAAAAAAAAUAAAADAAAALAAAAAQAAAF0AA
EAAAAAAG4AAwABAAAALAADAAoAAAF0AAQAQgAAAAYABAABAALoCegT//8AAOgB6BD//wAAAAAAAQ
AGABYAAAABAAIAAwAEAAUABgAHAAgACQAKAAsADAANAAABBgAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAMAAAAAACsAAAAAAAAAA0AAOgBAADoAQAAAAEAAOgCAADoAgAAAAIAAOgDAADoAw
AAAAMAAOgEAADoBAAAAAQAAOgFAADoBQAAAAUAAOgGAADoBgAAAAYAAOgHAADoBwAAAAcAAOgIAA
DoCAAAAAgAAOgJAADoCQAAAAkAAOgQAADoEAAAAAoAAOgRAADoEQAAAAsAAOgSAADoEgAAAAwAAO
gTAADoEwAAAA0AAwAA//kDWgLEAA8AHwAvADdANCgBBAUIAAIAAQJHAAUABAMFBGAAAwACAQMCYA
ABAAABVAABAQBYAAABAEwmNSY1JjMGBRorJRUUBgchIiYnNTQ2NyEyFgMVFAYnISImJzU0NhchMh
YDFRQGIyEiJic1NDYXITIWA1kUEPzvDxQBFg4DEQ8WARQQ/O8PFAEWDgMRDxYBFBD87w8UARYOAx
EPFmRHDxQBFg5HDxQBFgEQSA4WARQPSA4WARQBDkcOFhYORw8WARQAAAAABQAA/2oD6ANSAB8AIg
AlADMAPABsQGkjAQAGHQEJACcgAgcFA0cMAQAACQUACV4ABQAHBAUHYAAEAAoIBApgAAgAAgsIAm
AABgYDWAADAwxIDQELCwFYAAEBDQFJNDQBADQ8NDw7OTY1MC8uLCkoJSQiIRoXDgwJBgAfAR4OBR
QrATIWFxEUBgchIiYnNSEiJicRNDY/AT4BOwEyFhcVNjMPATMBBzMXNzUjFRQGByMRITU0NgERIx
UUBicjEQOyFx4BIBb96RceAf7RFx4BFhDkDzYW6BceASYhR6en/punp22w1h4X6QEeFgIm1x4X6A
J8IBb9WhceASAWoCAWAXcWNg/kEBYgFrcXd6cBfafCsOnpFh4B/puPFjb+TgKD6BYgAf6aAAAJAA
D/+QPoAwsADwAfAC8APwBPAF8AbwB/AI8AT0BMEQ0CBxAMAgYDBwZgDwkCAw4IAgIBAwJgCwUCAQ
AAAVQLBQIBAQBYCgQCAAEATI6LhoN+e3ZzbmtmY15bVlNOSzU1NTU1NTU1MxIFHSslFRQGByMiJi
c1NDYXMzIWExUUBicjIiYnNTQ2NzMyFgEVFAYHIyImJzU0NhczMhYBFRQGKwEiJic1NDY7ATIWAR
UUBicjIiYnNTQ2NzMyFgEVFAYHIyImPQE0NhczMhYBFRQGKwEiJic1NDY7ATIWARUUBicjIiY9AT
Q2NzMyFhMVFAYrASImPQE0NjsBMhYBHiAWshceASAWshceASAWshceASAWshceAWYgFrIXHgEgFr
IXHv6cIBayFx4BIBayFx4BZiAWshceASAWshceAWYgFrIWICAWshce/pwgFrIXHgEgFrIXHgFmIB
ayFiAgFrIXHgEgFrIWICAWshcemmwWHgEgFWwWIAEeAQZrFiABHhdrFx4BIP7NbBYeASAVbBYgAR
4CJGsWICAWaxYgIP7MaxYgAR4XaxceASD+zWwWHgEgFWwWIAEeAiRrFiAgFmsWICD+zGsWIAEeF2
sXHgEgAQhrFiAgFmsWICAAAAMAAP+5BBYCugAUACQAOQAeQBsuEQIAAQFHAwEBAAFvAgEAAGY1NC
gnFxIEBRYrJQcGIicBJjQ3ATYyHwEWFA8BFxYUAQMOAS8BLgE3Ez4BHwEeAQkBBiIvASY0PwEnJj
Q/ATYyFwEWFAFYHAUOBv78BgYBBAUQBBwGBtvbBgFE0AIOBiIIBgHRAgwHIwcIAWz+/AYOBhwFBd
```

vbBQUcBg4GAQQFRRwFBQEFBQ4GAQQGBhwFEATc2wYOAk79LwcIAwkDDAgC0AgGAQoCDv6P/vsFBR
wGDgbb3AUOBhwGBv78BRAAAAMAAP+xA30DCwAIABgAVQBOQEtKAQgHHxsCAAMAAQEAMRECAgEERw
AHCAdvAAgDCG8GAQMAA28AAAEAbwAEAgRwAAECAgFUAAEBAlgFAQIBAkwvLBUkPyY1ExIJBR0rNz
QuAQ4BHgE2ExEUBgcjIiYnETQ2FzMyFgUUBxYVFgcWBwYHFgcGByMiLgEnJiciJicRND4CNzY3Pg
I3PgMzMh4EBhcUDgEHDgIHMzIWjxYdFAEWHRRaFBCgDxQBFg6gDxYClB8JARkJCQkWBSAkSkglVj
IqRRMPFAEUGzocJhIKDgYFBAYQFQ8ZKhgUCAYCAgwIDAEIBAObK0BkDxQBFh0UARYBLP6bDxQBFg
4BZQ4WARQPMCMZEioiHyMfFT4nKwESDg8YARYOAWUOFgFAIzESCiIUGBYYIhYMEhoYIBINFSwWFA
QMDgZAAAAACwAA/7EDWQNSAA8AHwAvAD8ATwBfAG8AfwCPAJ8ArwD6QPesARITnAEODzkBDQ6MLQ
IKC3wBBgdsAQIDBkcAFAAUcCkBEiYBERASEWAoLgITJwEQDxMQYCUBDiIBDQwODWAkLQIPIwEMCw
8MYCEBCh4BCQgKCWAgLAILHwEIBwsIYB0BBhoBBQQGBWAcKwIHGwEEAwcEYBkBAhYBAQACAWAYKg
IDFwEAFAMAYAAVFQwVSUBAMDAgIBAQAACvraqpqKakop+dmpmYlpSSj42KiYiGhIJ/fXp5eHZ0cm
9tamloZmRiXltWU0BPQE5MSkdFQ0EwPzA+PDs3NTMxIC8gLywqJyUjIRAfEB4cGhcVExEADwAOIy
IhLwUXKzcVIyI9ASMiPQE0OwE1NDM3FSMiPQEjIj0BNDsBNTQzNxUjIj0BIyI9ATQ7ATU0MzcVIy
I9ASMiPQE0OwE1NDM3FSMiPQEjIj0BNDsBNTQzJREUBiMhIiYnETQ2NyEyFhMVFCsBFRQrATUzMh
0BMzI1FRQrARUUKwE1MzIdATMyNRUUKwEVFCsBNTMyHQEzMjUVFCsBFRQrATUzMh0BMzI1FRQrAR
UUKwE1MzIdATMyaz4JGwkJGwk+PgkbCQkbCT4+CRsJCRsJPj4JGwkJGwk+PgkbCQkbCQKdHhf+Lx
YeASAVAdEWII4JGwg/PwgbCQkbCD8/CBsJCRsIPz8IGwkJGwg/PwgbCQkbCD8/CBsJiEgJCQkSCQ
kJj0gJCQkSCQkJjkcJCQkSCQkIj0cJCQkSCAkJj0cJCQkRCQkJWfzLFiAgFgM1Fx4BIP07EgkJCU
gJCYYSCQkJSAkJhhIJCQlHCAmGEgkJCUcJCYYRCQkJRwkJAAAEAAD/ZgPzA1IADwAeACwAOQDhS7
AJUFhADwQBAQAFAQUCGhkCAwUDRxtLsApQWEAPBAEBAAUBBQQaGQIDBQNHG0APBAEBAAUBBQIaGQ
IDBQNHWVlLsAlQWEAkAAEAAgABAm0HBAICBQACBWsABQMABQNrBgEAAAxIAAMDDQNJG0uwClBYQC
oAAQACAAECbQACBAACBGsHAQQFAAQFawAFAwAFA2sGAQAADEgAAwMNA0kbQCQAAQACAAECbQcEAg
IFAAIFawAFAwAFA2sGAQAADEgAAwMNA0lZWUAXLi0BADQzLTkuOSYlIB8KCQAPAQ8IBRQrAQYHBg
cXPgIXBSYnJicmBQYHBhUUFx4BFzcGLgEnAQUeAQYHAxY2NzY3PgElIg4BFB4BMj4BNC4BAfJwZW
dHmhNSbDoBniAxMkFz/dcoFBY4N8F3gDlwWBgCuf7mJh0UIuJIjUFqQj8Z/gIuTS4uTVxNLi5NA1
IBMDFY7TdRKAYWQDQ2JkPiPEVGS3tsaYwS/AseSjUBFA8scHIv/qUFISY9Z2TtZS5NXE0uLk1cTS
4AAAAAAv/9/7EDXwMLABAAHQArQCgAAwQBAAEDAGAAAQICAVQAAQECWAACAQJMAQAbGhUUCQgAEA
EQBQUUKwEiDgMeAj4DNC4CARQOASIuAj4BMh4BAa1JhGA4AjxciI6GXjo6XoYBZXLG6MhuBnq89L
p+AsM4YISShF48BDRmfJp8aDD+n3XEdHTE6sR0dMQAAAAAAv///2oDoQMNAAgAIQArQCgfAQEADg
EDAQJHAAQAAAEEAGAAAQADAgEDYAACAg0CSRcjFBMSBQUZKwE0LgEGFBY+AQEUBiIvAQYjIi4CPg
QeAhcUBxcWAoOS0JKS0JIBHiw6FL9ke1CSaEACPGyOpI5sPAFFvxUBgmeSApbKmAaM/podKhW/RT
5qkKKObjoEQmaWTXtkvxUAAAAAAv/9/2oDWQNSACYATQA8QDlFQj8NBwUGAAFLSEY+DgUDACIaAg
IDA0cAAAEDAQADbQABAQxIAAMDAlgAAgINAkksKyAeFxIEBRYrET4BNzYXNjc1PgEyFhcTNhceAQ
cOAQcOAgcVFAYHISImJzU0LgE3HgIXITU+ATc+AT8BMjY3NicuAQ4BBxEuAScOAQcVJgcmBgcmBg
JKSTNEGSACRmtEBQFeTDc2FxdwFRciUhEmGf6lGiQDHBY+AhYcAQFbEG4NFUIWRQQGAQQNFkg8WB
YCIhwYIgMxOhpCDj46AaM8TAQrChAGazVMSDn+7y0cE3Y4FhALDipMFpsZJAMmGqochHQdN2x6Fw
MmYhMZIAQNAgQVGiMOFiIDAW0bJAICJBu/MTsQEhsJOAAAAgAA/74CygMLAAUAIgAyQC8UBQMCBA
IAAUcDAQIAAnAEAQEAAAFUBAEBAQBWAAABAEoHBhgWEhAGIgchEAUFFSsBIREBHwETMhceARcRFA
YHBiMiLwEHBiMiJy4BNRE0Njc2MwKD/cQBHjLsBwwMExQBFhIKDhsU9vYUGg0MEhYWEgwNAsP9Sw
ESL+MC/QUIHhT9MRMgBwQS7OwTBQcgEwLPEyAHBQAABgAA/2oDWQNSABMAGgAjADMAQwBTAHJAbx
QBAgQsJAIHBkA4AggJUEgCCgsERwACAAMGAgNgAAYABwkGB2ANAQkACAsJCGAOAQsACgULCmAABA
QBWAABAQxIDAEFBQBYAAAADQBJREQ0NBsbRFNEUkxKNEM0Qjw6MC4oJhsjGyMTJhQ1Ng8FGSsBHg
EVERQGByEiJicRNDY3ITIWFwcVMyYvASYTESMiJic1IRETNDYzITIWHQEUBiMhIiY1BTIWHQEUBi
MhIiY8ATQ2MwUyFh0BFAYjISImPQE0NjMDMxAWHhf9EhceASAWAfQWNg9K0gUHrwbG6BceAf5Tjw
oIAYkICgoI/ncICgGbCAoKCP53CAoKCAGJCAoKCP53CAoKCAJ+EDQY/X4XHgEgFgN8Fx4BFhAm0h
EGrwf8sAI8IBXp/KYB4wcKCgckCAoKCFkKCCQICgoIJAgKjwoIJAgKCggkCAoAAAAAA//9/7EDXw
MLAA8ANwBEAEhARSkBBQMJAQIBAAJHAAQCAwIEA20AAwUCAwVrAAcAAgQHAmAABQAAAQUAYAABBg
YBVAABAQZYAAYBBkwVHisTFiYmIwgFHCslNTQmKwEiBh0BFBY7ATI2EzQuASMiBwYfARYzMjc+AT
IWFRQGBw4BFxUUFjsBMjY0Nj8BPgMXFA4BIi4CPgEyHgEB9AoIawgKCghrCAqPPlwxiEcJDUoEBg
kFHiU4KhYbIzwBCghrCAoYEhwKHhQM13LG6MhuBnq89Lp+UmsICgoIawgKCgF/MVQudw0LNwQHJh
seEhUaDA9CJRQICgoSIgsQBhocKFJ1xHR0xOrEdHTEAAEAAAABAACCKpnPXw889QALA+gAAAA2E
iuQQAAAADYSK5B//3/ZgQWA1IAAAAIAAIAAAAAAAAAAQAAA1L/agAABC///f/0BBYAAQAAAAAAAA
AAAAAAAAAAAA4D6AAAA1kAAAPoAAAD6AAABC8AAAOgAAADWQAAA+gAAANZ//0DoP//A03//QLKAA
ADWQAAA1n//QAAAAAAZgD6AegCWgMABEgFHAVkBbIGSAacB1AH1gAAAAEAAAAOALAACwAAAAAAg
BeAG4AcwAAAQsLcAAAAAAAAAASAN4AAQAAAAAAAAA1AAAAAQAAAAAAAQAIADUAAQAAAAAAgAHAD
0AAQAAAAAAAwAIAEQAAQAAAAAABAAIAEwAAQAAAAAABQALAFQAAQAAAAAABgAIAF8AAQAAAAAACg
ArAGcAAQAAAAACwATAJIAAwABBAkAAABqAKUAAwABBAkAAQAQAQ8AAwABBAkAAgAOAR8AAwABBA

kAAwAQAS0AAwABBAkABAAQAT0AAwABBAkABQAWAU0AAwABBAkABgAQAWMAAwABBAkACgBWAXMAAw
ABBAkACwAmAclDb3B5cmlnaHQgKEMpIDIwMTggYnkgb3JpZ2luYWwgYXV0aG9ycyBAIGZvbnRlbG
xvLmNvbWZvbnRlbGxvUmVndWxhcmZvbnRlbGxvZm9udGVsbG9WZXJzaW9uIDEuMGZvbnRlbGxvR2
VuZXJhdGVkIGJ5IHN2ZzJ0dGYgZnJvbSBGb250ZWxsbyBwcm9qZWN0Lmh0dHA6Ly9mb250ZWxsby
5jb20AQwBvAHAAeQByAGkAZwBoAHQAIAAoAEMAKQAgADIAMAAxADgAIABiAHkAIABvAHIAaQBnAG
kAbgBhAGwAIABhAHUAdABoAG8AcgBzACAAQAAgAGYAbwBuAHQAZQBsAGwAbwAuAGMAbwBtAGYAbw
BuAHQAZQBsAGwAbwBSAGUAZwB1AGwAYQByAGYAbwBuAHQAZQBsAGwAbwBmAG8AbgB0AGUAbABsAG
8AVgBlAHIAcwBpAG8AbgAgADEALgAwAGYAbwBuAHQAZQBsAGwAbwBHAGUAbgBlAHIAYQB0AGUAZA
AgAGIAeQAgAHMAdgBnADIAdAB0AGYAIABmAHIAbwBtACAARgBvAG4AdABlAGwAbABvACAAcAByAG
8AagBlAGMAdAAuAGgAdAB0AHAAOgAvAC8AZgBvAG4AdABlAGwAbABvAC4AYwBvAG0AAAAAAgAAAA
AAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAOAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4BDw
AEbWVudQRkb2NzAnRoBGNvZGUNdGh1bWJzLXVwLWFsdAltaWNyb2NoaXAGY2hyb21lC2NpcmNsZS
10aGluCHNlYXJjaC0xB3VwLWhhbmQOYm9va21hcmstZW1wdHkIZG9jLXRleHQMaGVscC1jaXJjbG
VkAAAAAQAB//8ADwAAAAAAAAAAAAAAAAAAAAAGAAYABgAGANS/2YDUv9msAAsILAAVVhFWSAgS7
gADlFLsAZTWliwNBuwKFlgZiCKVViwAiVhuQgACABjYyNiGyEhsABZsABDI0SyAAEAQ2BCLbABLL
AgYGYtsAIsIGQgsMBQsAQmWrIoAQpDRWNFUltYISMhG4pYILBQUFghsEBZGyCwOFBYIbA4WVkgsQ
EKQ0VjRWFksChQWCGxAQpDRWNFILAwUFghsDBZGyCwwFBYIGYgiophILAKUFhgGyCwIFBYIbAKYB
sgsDZQWCGwNmAbYFlZWRuwAStZWSOwAFBYZVlZLbADLCBFILAEJWFkILAFQ1BYsAUjQrAGI0IbIS
FZsAFgLbAELCMhIyEgZLEFYkIgsAYjQrEBCkNFY7EBCkOwAWBFY7ADKiEgsAZDIIogirABK7EwBS
WwBCZRWGBQG2FSWVgjWSEgsEBTWLABKxshsEBZI7AAUFhlWS2wBSywB0MrsgACAENgQi2wBiywBy
NCIyCwACNCYbACYmawAWOwAWCwBSotsAcsICBFILALQ2O4BABiILAAUFiwQGBZZrABY2BEsAFgLb
AILLIHCwBDRUIqIbIAAQBDYEItsAkssABDI0SyAAEAQ2BCLbAKLCAgRSCwASsjsABDsAQlYCBFii
NhIGQgsCBQWCGwABuwMFBYsCAbsEBZWSOwAFBYZVmwAyUjYUREsAFgLbALLCAgRSCwASsjsABDsA
QlYCBFiiNhIGSwJFBYsAAbsEBZI7AAUFhlWbADJSNhRESwAWAtsAwsILAAI0KyCwoDRVghGyMhWS
ohLbANLLECAkWwZGFELbAOLLABYCAgsAxDSrAAUFggsAwjQlmwDUNKsABSWCCwDSNCWS2wDywgsB
BiZrABYyC4BABjiiNhsA5DYCCKYCCwDiNCIy2wECxLVFixBGREWSSwDWUjeC2wESxLUVhLU1ixBG
REWRshWSSwE2UjeC2wEiyxAA9DVVixDw9DsAFhQrAPK1mwAEOwAiVCsQwCJUKxDQIlQrABFiMgsA
MlUFixAQBDYLAEJUKKiiCKI2GwDiohI7ABYSCKI2GwDiohG7EBAENgsAIlQrACJWGwDiohWbAMQ0
ewDUNHYLACYiCwAFBYsEBgWWawAWMgsAtDY7gEAGIgsABQWLBAYFlmsAFjYLEAABMjRLABQ7AAPr
IBAQFDYEItsBMsALEAAkVUWLAPI0IgRbALI0KwCiOwAWBCIGCwAWG1EBABAA4AQkKKYLESBiuwci
sbIlktsBQssQATKy2wFSyxARMrLbAWLLECEystsBcssQMTKy2wGCyxBBMrLbAZLLEFEystsBossQ
YTKy2wGyyxBxMrLbAcLLEIEystsB0ssQkTKy2wHiwAsA0rsQACRVRYsA8jQiBFsAsjQrAKI7ABYE
IgYLABYbUQEAEADgBCQopgsRIGK7ByKxsiWS2wHyyxAB4rLbAgLLEBHistsCEssQIeKy2wIiyxAx
4rLbAjLLEEHistsCQssQUeKy2wJSyxBh4rLbAmLLEHHistsCcssQgeKy2wKCyxCR4rLbApLCA8sA
FgLbAqLCBgsBBgIEMjsAFgQ7ACJWGwAWCwKSohLbArLLAqK7AqKi2wLCwgIEcgILALQ2O4BABiIL
AAUFiwQGBZZrABY2AjYTgjIIpVWCBHICCwC0NjuAQAYiCwAFBYsEBgWWawAWNgI2E4GyFZLbAtLA
CxAAJFVFiwARawLCqwARUwGyJZLbAuLACwDSuxAAJFVFiwARawLCqwARUwGyJZLbAvLCA1sAFgLb
AwLACwAUVjuAQAYiCwAFBYsEBgWWawAWOwASuwC0NjuAQAYiCwAFBYsEBgWWawAWOwASuwABa0AA
AAAABEPiM4sS8BFSotsDEsIDwgRyCwC0NjuAQAYiCwAFBYsEBgWWawAWNgsABDYTgtsDIsLhc8Lb
AzLCA8IEcgsAtDY7gEAGIgsABQWLBAYFlmsAFjYLAAQ2GwAUNjOC2wNCyxAgAWJSAuIEewACNCsA
IlSYqKRyNHI2EgWGIbIVmwASNCsjMBARUUKi2wNSywABawBCWwBCVHI0cjYbAJQytlii4jICA8ij
gtsDYssAAWsAQlsAQlIC5HI0cjYSCwBCNCsAlDKyCwYFBYILBAUVizAiADIBuzAiYDGllCQiMgsA
hDIIojRyNHI2EjRmCwBEOwAmIgsABQWLBAYFlmsAFjYCCwASsgiophILACQ2BkI7ADQ2FkUFiwAk
NhG7ADQ2BZsAMlsAJiILAAUFiwQGBZZrABY2EjICCwBCYjRmE4GyOwCENGsAIlsAhDRyNHI2FgIL
AEQ7ACYiCwAFBYsEBgWWawAWNgIyCwASsjsARDYLABK7AFJWGwBSWwAmIgsABQWLBAYFlmsAFjsA
QmYSCwBCVgZCOwAyVgZFBYIRsjIVkjICCwBCYjRmE4WS2wNyywABYgICCwBSYgLkcjRyNhIzw4Lb
A4LLAAFiCwCCNCICAgRiNHsAErI2E4LbA5LLAAFrADJbACJUcjRyNhsABUWC4gPCMhG7ACJbACJU
cjRyNhILAFJbAEJUcjRyNhsAYlsAUlSbACJWG5CAAIAGNjIyBYYhshWWO4BABiILAAUFiwQGBZZr
ABY2AjLiMgIDyKOCMhWS2wOiywABYgsAhDIC5HI0cjYSBgsCBgZrACYiCwAFBYsEBgWWawAWMjIC
A8ijgtsDssIyAuRrACJUZSWCA8WS6xKwEUKy2wPCwjIC5GsAIlRlBYIDxZLrErARQrLbA9LCMgLk
awAiVGUlggPFkjIC5GsAIlRlBYIDxZLrErARQrLbA+LLA1KyMgLkawAiVGUlggPFkusSsBFCstsD
8ssDYriiAgPLAEI0KKOCMgLkawAiVGUlggPFkusSsBFCuwBEMusCsrLbBALLAAFrAEJbAEJiAuRy
NHI2GwCUMrIyA8IC4jOLErARQrLbBBLLEIBCVCsAAWsAQlsAQlIC5HI0cjYSCwBCNCsAlDKyCwYF
BYILBAUVizAiADIBuzAiYDGllCQiMgR7AEQ7ACYiCwAFBYsEBgWWawAWNgILABKyCKimEgsAJDYG
QjsANDYWRQWLACQ2EbsANDYFmwAyWwAmIgsABQWLBAYFlmsAFjYbACJUZhOCMgPCM4GyEgIEYjR7
ABKyNhOCFZsSsBFCstsEIssDUrLrErARQrLbBDLLA2KyEjICA8sAQjQiM4sSsBFCuwBEMusCsrLb
BELLAAFSBHsAAjQrIAAQEVFBMusDEqLbBFLLAAFSBHsAAjQrIAAQEVFBMusDEqLbBGLLEAARQTsD

```
IqLbBHLLA0Ki2wSCywABZFIyAuIEaKI2E4sSsBFCstsEkssAgjQrBIKy2wSiyyAABBKy2wSyyyAA
FBKy2wTCyyAQBBKy2wTSyyAQFBKy2wTiyyAABCKy2wTyyyAAFCKy2wUCyyAQBCKy2wUSyyAQFCKy
2wUiyyAAA+Ky2wUyyyAAE+Ky2wVCyyAQA+Ky2wVSyyAQE+Ky2wViyyAABAKy2wVyyyAAFAKy2wWC
yyAQBAKy2wWSyyAQFAKy2wWiyyAABDKy2wWyyyAAFDKy2wXCyyAQBDKy2wXSyyAQFDKy2wXiyyAA
A/Ky2wXyyyAAE/Ky2wYCyyAQA/Ky2wYSyyAQE/Ky2wYiywNysusSsBFCstsGMssDcrsDsrLbBkLL
A3K7A8Ky2wZSywABawNyuwPSstsGYssDgrLrErARQrLbBnLLA4K7A7Ky2waCywOCuwPCstsGkssD
grsD0rLbBqLLA5Ky6xKwEUKy2wayywOSuwOystsGwssDkrsDwrLbBtLLA5K7A9Ky2wbiywOisusS
sBFCstsG8ssDorsDsrLbBwLLA6K7A8Ky2wcSywOiuwPSstsHIsswkEAgNFWCEbIyFZQiuwCGWwAy
RQeLABFTAtAEu4AMhSWLEBAY5ZsAG5CAAIAGNwsQAFQrIAAQAqsQAFQrMKAgEIKrEABUKzDgABCC
qxAAZCugLAAAEACSqxAAdCugBAAAEACSqxAwBEsSQBiFFYsECIWLEDZESxJgGIUVi6CIAAAQRAiG
NUWLEDAERZWVlZswwCAQwquAH/hbAEjbECAEQAAA==') format('truetype');
    }
    #sidebar-treeview #main-treeview .icon {
        font-family: 'fontello';
        font-size: 16px;
    }
    #header-section.main-header .icon-menu::before {
        content: '\e801';
        font-family: 'fontello';
        font-size: 27px;
    }
    #sidebar-treeview #main-treeview .icon-microchip::before {
        content: '\e806';
    }
    #sidebar-treeview #main-treeview .icon-thumbs-up-alt::before {
        content: '\e805';
    }
    #sidebar-treeview #main-treeview .icon-docs::before {
        content: '\e802';
    }
    #sidebar-treeview #main-treeview .icon-th::before {
        content: '\e803';
    }
    #sidebar-treeview #main-treeview .icon-code::before {
        content: '\e804';
    }
    #sidebar-treeview #main-treeview .icon-chrome::before {
        content: '\e807';
    }
    #sidebar-treeview #main-treeview .icon-up-hand::before {
        content: '\e810';
    }
    #sidebar-treeview #main-treeview .icon-bookmark-empty::before {
        content: '\e811';
    }
    #sidebar-treeview #main-treeview .icon-help-circled::before {
        content: '\e813';
    }
    #sidebar-treeview #main-treeview .icon-doc-text::before {
        content: '\e812';
    }
    #sidebar-treeview #main-treeview .icon-circle-thin::before {
        content: '\e808';
    }
    .e-sidebar.e-open #main-treeview .e-text-content {
        padding: 0 0 0 24px;
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    .e-sidebar.e-close #main-treeview .e-text-content {
        padding: 0 0 0 8.5px;
    }
    #main-treeview ul {
        overflow: inherit;
    }
    /* end of icon styles */
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/treeview-cs1" %}

Fixed sidebar in Vue Sidebar component

The Sidebar does not require any specific style to make it as a fixed one. By default, the Sidebar position will be in fixed state. The following example demonstrates that the Sidebar is rendered with a fixed position. The position of the Sidebar will not change when scrolling the main content area.

APP.VUE

```
<template>
    <div id="app">
       <!-- Declaration of sidebar element-->
        <div class="col-lg-8 control-section">
            <ejs-sidebar  id="default-sidebar" :width='width' >
                <div class="sidebar-header header-cover" style="background-
color:#0378d5">
                    <div class="image-container">
                        <div class="sidebar-image">
                        </div>
                    </div>
                    <div style="padding: 0 0 5px 0;">
                        <a class="sidebar-brand" href="#settings-dropdown">
                            john.doe@gmail.com
                        </a>
                        <span class="sf-icon-down icon"></span>
                    </div>
                </div>
                <!-- Sidebar navigation -->
                <ul class="nav sidebar-nav">
                    <li>
                        <a href="#">
                            <i class="sf-icon-sidebar sf-icon-file"></i>
                            <span class="e-text"> Inbox</span>
                        </a>
                    </li>
                    <li>
                        <a href="#">
                            <i class="sf-icon-sidebar sf-icon-starred"></i>
                            <span class="e-text"> Starred</span>
                        </a>
                    </li>
                    <li>
                        <a href="#">
                            <i class="sf-icon-sidebar sf-icon-recent"></i>
                            <span class="e-text">Snoozed</span>
                        </a>
                    </li>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <li>
                        <a href="#">
                            <i class="sf-icon-sidebar sf-icon-
important"></i>
                            <span class="e-text">Important</span>
                        </a>
                    </li>
                    <li>
                        <a href="#">
                            <i class="sf-icon-sidebar sf-icon-offline"></i>
                            <span class="e-text"> Sent</span>
                        </a>
                    </li>
                    <li>
                        <a href="#">
                            <i class="sf-icon-sidebar sf-icon-backup"></i>
                            <span class="e-text"> Draft</span>
                        </a>
                    </li>
                </ul>
            </ejs-sidebar>
        <!-- End of sidebar element-->
        <!-- Main content Declaration-->
            <div class="e-content-animation">
                <div class="content">
                    <div id="left">
                        <span id="hamburger" class="e-icons menu
default"></span>
                    </div>
                    <div id="center">
                        <span>Inbox</span>
                    </div>
                    <div id="right">
                        <span class="sf-icon-search"></span>
                    </div>
                </div>
                <div>
                    <div class="e-control e-listview e-list-template e-
touch">
                        <ul class="e-list-parent e-ul ">
                            <li class="e-list-group-item e-level-1"
role="group" data-uid="group-list-item-Today"
                                aria-level="1">
                                <div class="e-text-content"
role="presentation"><span class style="width: 100%; margin-left: 2%; margin-
top: -2%;">Today</span></div>
                            </li>
                            <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                    <span class="e-avatar e-icon sf-icon-
profile"></span>
                                    <div>
                                        <span class="e-list-item-header e-
name">Albert Lives</span>
                                    </div>
```

```
                                    <span class="received e-list-content e-
second-heading">Opening for Sales Manager</span>
                                    <span class="e-list-item-header sf-icon-
star">
                                        <span class="e-list-text">Hello Uta
Morgan,</span></span>
                                </div>
                            </li>
                            <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                    <span class="e-avatar e-icon sf-icon-
profile"></span>
                                    <div>
                                        <span class="e-list-item-header e-
name">
                                            Ila Russo</span>
                                    </div>
                                    <span class="received e-list-content e-
second-heading">Business dinner invitation
                                    </span>
                                    <span class="e-list-item-header sf-icon-
star">
                                        <span class="e-list-text">Hello
Jelani Moreno,</span></span>
                                </div>
                            </li>
                            <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                    <span class="e-avatar e-icon sf-icon-
profile"></span>
                                    <div>
                                        <span class="e-list-item-header e-
name">
                                            Garth Owen</span>
                                    </div>
                                    <span class="received e-list-content e-
second-heading">Application for Job Title</span>
                                    <span class="e-list-item-header sf-icon-
star">
                                        <span class="e-list-text">Hello Ila
Russo,</span></span>
                                </div>
                            </li>
                            <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                    <span class="e-avatar e-icon sf-icon-
profile"></span>
                                    <div>
                                        <span class="e-list-item-header e-
name">Ursula Patterson</span>
                                    </div>
                                    <span class="received e-list-content e-
second-heading">Hello Kerry Best,</span>
```

```
                                    <span class="e-list-item-header sf-icon-
star">
                                        <span class="e-list-text">Programmer
Position Application</span></span>
                                </div>
                            </li>
                            <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                    <span class="e-avatar e-icon sf-icon-
profile"></span>
                                    <div>
                                        <span class="e-list-item-header e-
name">
                                            Nichole Rivas</span>
                                    </div>
                                    <span class="received e-list-content e-
second-heading">Annual Conference</span>
                                    <span class="e-list-item-header sf-icon-
star">
                                        <span class="e-list-text">Hi Igor
Mccoy,</span></span>
                                </div>
                            </li>
                            <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                    <span class="e-avatar e-icon sf-icon-
profile"></span>
                                    <div>
                                        <span class="e-list-item-header e-
name">
                                            Nichole Rivas</span>
                                    </div>
                                    <span class="received e-list-content e-
second-heading">Annual Conference</span>
                                    <span class="e-list-item-header sf-icon-
star">
                                        <span class="e-list-text">Hi Igor
Mccoy,</span></span>
                                </div>
                            </li>
                            <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                    <span class="e-avatar e-icon sf-icon-
profile"></span>
                                    <div>
                                        <span class="e-list-item-header e-
name">
                                            Nichole Rivas</span>
                                    </div>
                                    <span class="received e-list-content e-
second-heading">Annual Conference</span>
                                    <span class="e-list-item-header sf-icon-
star">
```

```
                                        <span class="e-list-text">Hi Igor
Mccoy,</span></span>
                                </div>
                            </li>
                            <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                    <span class="e-avatar e-icon sf-icon-
profile"></span>
                                    <div>
                                        <span class="e-list-item-header e-
name">
                                            Nichole Rivas</span>
                                    </div>
                                    <span class="received e-list-content e-
second-heading">Annual Conference</span>
                                    <span class="e-list-item-header sf-icon-
star">
                                        <span class="e-list-text">Hi Igor
Mccoy,</span></span>
                                </div>
                            </li>
                            <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                    <span class="e-avatar e-icon sf-icon-
profile"></span>
                                    <div>
                                        <span class="e-list-item-header e-
name">
                                            Nichole Rivas</span>
                                    </div>
                                    <span class="received e-list-content e-
second-heading">Annual Conference</span>
                                    <span class="e-list-item-header sf-icon-
star">
                                        <span class="e-list-text">Hi Igor
Mccoy,</span></span>
                                </div>
                            </li>
                            <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                    <span class="e-avatar e-icon sf-icon-
profile"></span>
                                    <div>
                                        <span class="e-list-item-header e-
name">Ursula Patterson</span>
                                    </div>
                                    <span class="received e-list-content e-
second-heading">Hello Kerry Best,</span>
                                    <span class="e-list-item-header sf-icon-
star">
                                        <span class="e-list-text">Programmer
Position Application</span></span>
                                </div>
                            </li>
```

```
                        </ul>
                    </div>
                </div>
            </div>
            <!--end of main content declaration -->
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SidebarPlugin } from '@syncfusion/ej2-vue-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SidebarPlugin);
export default {
       data () {
        return {
            width : '290px',
        };
     },
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
.control-section {
    overflow: hidden;
  }

  .e-listview .e-list-parent {
    margin: 0 ;
    padding: 15px 0 15px 0 ;
  }

  .e-listview .e-list-group-item {
    background-color: #ffffff ;
  }

  .e-second-heading {
    color: rgba(0, 0, 0, 0.64) ;
    display: block;
    font-size: 15px ;
    font-weight: 500 ;
  }

  .e-listview.e-list-template .e-list-wrapper.e-list-multi-line.e-list-
avatar .e-avatar {
    top: 1.5666em;
    border-radius: 50%;
    border-color: #4f4b4b ;
    border: 1px solid #BADA55;
  }

  .e-listview {
    overflow: initial;
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.sf-icon-profile:before {
  float: right;
  content: '\e717';
  font-size: 25px;
}

.e-avatar {
  background: none;
  color: #000;
}

.e-today {
  height: 50px;
  padding: 15px;
}

.e-today span {
  border-color: rgba(0, 0, 0, 0.12);
  color: rgba(0, 0, 0, 0.54);
  font-size: 15px;
  font-weight: 600;
  height: 36px;
}

.content {
  height: 47px;
  background: #0378d5;
  width: 100%;
}

#left {
  float: left;
  width: 30px;
  height: 47px;
}

#center {
  display: inline-block;
  margin: 0 auto;
  width: 0px;
  height: 47px;
  padding: 12px 0 12px 0;
  font-size: 18px;
  color: #ffff;
  float: left;
  text-align: center;
  padding-left: 40px;
}

#right {
  float: right;
  width: 28px;
  height: 47px;
  padding: 16px 0 0 0;
  color: #ffff;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}

.sf-icon-file:before {
  content: '\e701';

}

.sf-icon-share:before {
  content: '\e70b';
}


.sf-icon-starred:before {
  content: '\e708';
}

.sf-icon-star:before {
  content: '\e707';
  font-size: 15px;
  color: #000;
  float: right;

}

.sf-icon-recent:before {
  content: '\e70c';
}

.sf-icon-important:before {
  content: '\e713';
}

.e-sidebar.e-left {
  border: none;
}

.sf-icon-offline:before {
  content: '\e714';
  font-size: 30px;
}

.sf-icon-backup:before {
  content: '\e712';

}

.sf-icon-down:before {
  content: '\e709';

}

.sf-icon-search:before {
  content: '\e711';
  font-size: 18px;

}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.sf-icon-sidebar {
  content: '\e701';
  color: #000000;
  display: inline-block;
  margin-right: 16px;
  width: 40px;
  text-align: left;
  font-size: 20px ;
  vertical-align: middle;
}

ul.nav.sidebar-nav {
  padding: 5px 0 5px 0;
  margin: 0px;
  max-height:200px;
  overflow-y: auto;
}

.sidebar-nav {
  list-style: none;
  padding: 5px 0 5px 0;
}

a.sidebar-brand {
  color: #ffffff;
  font-size: 15px;
  text-decoration: none;
}

.nav.sidebar-nav li:hover {
  text-decoration: none;
  background-color: #eee;
}

span.sf-icon-down.icon {
  font-size: 12px;
}

.sidebar-nav li a {
  position: relative;
  cursor: pointer;
  user-select: none;
  display: block;
  height: 48px;
  line-height: 48px;
  padding: 0 56px 0 20px;
  text-decoration: none;
  font-weight: 500;
  overflow: hidden;
  font-size: 14px;

}

/* sample-level styles */
.image-container {
  padding: 18px 0 0 16px;
```

```
  }

  .sidebar-image {
    background:
url(https://ej2.syncfusion.com/showcase/angular/expensetracker/assets/images
/user.svg) no-repeat scroll 0 0 transparent;
    background-position-x: left;
    height: 85px;
    width: 100%;
    background-size: 85px;
    /* margin-left: 20px; */
  }

  .sidebar-brand {
    height: 48px;
    line-height: 48px;
    padding-left: 20px;
    padding-right: 40px;
    text-decoration: none;
    font-weight: 500;
    overflow: hidden;
    display: inline-flex;
  }

  .sidebar-image img {
    width: 54px;
    height: 54px;
    margin: 16px;
    border-radius: 50%;
    -webkit-transition: all 0.2s ease-in-out;
    -o-transition: all 0.2s ease-in-out;
    transition: all 0.2s ease-in-out;
  }

  #default-sidebar {
    color: #ffffff;
  }

  /* sample-level styles */

  #hamburger.menu {
    font-size: 23px;
    cursor: pointer;
    float: left;
    line-height: 50px;
    position: absolute;
    z-index: 1000;
    margin-left: 25px;
    color: #ffff;
  }

  #hamburger.menu:before {
    content: '\e10d';
  }

  #hamburger.menu.e-rtl {
    position: relative;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    float: right;
  }

  @font-face {
  font-family: 'e-icons';
  src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAkAAAAg4aGVhZBLt+DYAAADQAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAAYAAAAA
wbG9jYQukCgIAAAIkAAAAGm1heHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAzoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAAAAADAABAAAAAQAAlbrm7l8
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAD8wPzAAAACAACAAAAAAAAAAAEAAAAMAQIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAAgAAAAMAAAAUAAMAAQA
AABQABABeAAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAgABwAKAAkABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAAAAAIAAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAAA+oD6gALAAATCQEXCQEHCQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAAD6gOkAAMABwALAAA3IRUhESEVIREhFSE
VA9b8KgPW/CoD1vwq6I0B64wB640AAAEAAAAAA+oD4QCaAAABMx8aHQEPDjEPAh8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBwYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHBoZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAwIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICAkJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBwgICAkKCgs
LDAwNDQ4ODxAQEBEWFxYWFhYVFRQUExIRERAOFxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEhERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTExUUFhUWFxYWFxEQEBAPDg4
NDQwMCwsKCgkICAgHBgYFBQQEBQQAAAAAAwAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8fFz8XLxcPBgI+BQQDAwMCAT8EBAM
DAwIBAQIDAwMEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAwMDAgEBAgMDAwQE0AUEAwMDAgEBAgMDAwQ
FfAUEAwMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEB0AECAwMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAwMDAgEBAgMDAwQFaAQEAwQCAgElERMVFhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxYVExEQDgsJCAUDAQEDBQcKCw4QERMVFhcZGxsdHh4fICAhIiIhICAfHh4dHBoZFxYVExE
QDgsKBwUDAQEDBQcKCw4AAAIAAAAAA9MD6QALAE8AAAEOAQcuASc+ATceAQEHBgcnJgYPAQYWHwE
GFBcHDgEfAR4BPwEWHwEeATsBMjY/ATY3FxY2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGAps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAwdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYMhhSBhIIrgg
FAzIfE4QJDAwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOECQwMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4zA+v+Ff4VHwHMAc0AAAAAAQAAAAADAQPqAAUAAAEXCQEHAQLlHf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAD8wPzAEAAgQEBAAAlDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUfHTsBPx09AS8dKwEPHQL1DQ0ODg4PDw8QEBAQERERERUUFBQTExITERERERBAPDw0ODAw
LCwkJCAcGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAgQDBQUGBgcHCQgJCv3QDQ0ODg4PDw8QEBA
QERERERUUFBQTExITERERERBAPDw0ODAwLCwkJCAcGBgQEAgL8fgIDBQUHCAkKCwwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcYFhYWFRQUExIREQ8PDg0MCwoJCAcFBQMCAgMFBQcICQoLDA0ODw8
RERITFBQVFhYWGBcYGRkZGhoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBAREREUEhMTFBQUFREREREQEBAQDw8PDg4ODQ31ERE
RERAQEBAPDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBAREREUEhM
TFBQUFRoZGRkYFxgWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBU
WFhYYFxgZGRkaGhkZGRgXGBYWFhUUFBMSERIPDw4NDAsKCQgHBQUDAgIDBQUHCAkKCwwNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgwMDAsKCggIBwU
FAwIBAQIDBQUHCAgKCgsMDAwODQ4PDxAQEREREhISExQUFRUWFhcXGBgZGRoaGxscHR0eHh8fICA
hIiIjIyQkJSUmJicoKCkqKisrLCwtLS4uLy8wMDEyMjM0NDU2Njc4ODk6Ozw9PT4/QEFCQ0RFRkd
ISUpLTE1OUFJUVldZWltdXl9hYmNlZmhqa21vcXN1dnd5ent9gIGDhYaIiouNjpCSlJeZm52foKK
jpqmrra+xs7W3ubu9v8HDxcfJy83P0dPV19nb3d/h4+Xn6evt7/Hz9ff5+/0AAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
BayBlLWljb25zLW1ldHJvUmVndWxhcmUtaWNvbnMtbWV0cm9lLWljb25zLW1ldHJvVmVyc2lvbiA
xLjBlLWljb25zLW1ldHJvRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwBlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQByAHMAaQBvAG4AIAAxAC4AMABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAB
NAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwBAgEDAQQBBQEGAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMHbWVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHQMY2hldnJvbi1sZWZ0CE1UX0NsZWFyDE1UX0p1bmttYWlscwR
zdG9wAAA=) format('truetype');
  font-weight: normal;
  font-style: normal;
  }

  @font-face {
    font-family: 'e-sb-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj0gSSgAAAEoAAAAVmNtYXDnt+g6AAAB6AAAAGZnbHlmMmV
udQAAAogAAAz0aGVhZBMA8GsAAADQAAAANmhoZWEHkAOAAAAArAAAACRobXR4Zan/8gAAAYAAAAB
obG9jYSiAJP4AAAJQAAAANm1heHABKwBwAAABCAAAACBuYW1lTGtTDAAAD3wAAAJJcG9zdLRM+38
AABHIAAABggABAAADUv9qAFoEAP/0//sD7QABAAAAAAAAAAAAAAAAAAAAAGgABAAAAAQAAoeBeDl8
PPPUACwPoAAAAANgA1c4AAAAA2ADVzv/0AAAD7QPqAAAACAACAAAAAAAAAAAEAAAAaAGQABwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPpAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnGANS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPo//Q
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+j//gP
oAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAAAAACAAAAAwAAABQAAwABAAA
AFAAEAFIAAAAEAAQAAQAA5xj//wAA5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcACAAJAAoACwA
MAA0ADgAPABAAEQASABMAFAAVABYAFwAZABgAAAAAAAAAVgCIANABBAE+AUwBYgGMAagBtgHOAmA
CsALUA24D2gRWBJAE0gVCBZgFuAXsBkwGegAAAAP/9AAAA+oDeQASACQALwAAEwYSFxY2NxcTBRc
OAScmAjcOATcRHgEXISc3NTYnAQYiJwEOATcTFjI3ATc2NSEWBhPrxoDeTV0+/mxgQKlhorsQBw1
lDB4RAbdq+QEN/tsHFAf+2wUEMv8HEQcBAAQC/dUBAonK/tYdD15cRwGwokpMSwsXAQi3GDGR/rg
SJRBRZNoNCP7bBgYBJQIMMf8ABwcBAAcEBAoAAAADAAAAAAPqA20ABAAJABkAAAkBESERJRUJATU
jER4BMyEyNjURNCYnIQ4BAfYBtvyVA2v+Sv5LPwEjGwNrGyMjG/yVGyMBdwFF/gICAHEl/rsBRyP
9jxsjIxsCcRsjAQEjAAAAAwAAAAD6gNtABQAGQApAAABJyYOARYfARUhNTc+AS4BDwERCQE1FQk
BNSMRHgEzITI2NRE0JichDgEDrNILGBAEC/b8lekLBBAYC8UBrQG+/kL+Uz8BIxsDaxsjIxv8lRs
jATaYCAQWGQiyKzWoCBkWBAiOAXf+wAFLbR/+tQFBKf2PGyMjGwJxGyMBASMABQAAAAAD7QNLAAQ
ABwAKAA0AFgAANyElByc3BRMBJQkCISUzNzYmByIGJwYDsP5tRUWZAYED/FABff5/AdwB2PxQAv6
4LAYRCwcIAaLaMjIE0gHu/hLSARz+uwFbGVgZEAECAgAAAAAFAAAAAAPqA64ABQAIAAsAEQAZAAA
lISUXPwEFLQENARElAScHASUFESERNyc1JQN6/QUBJAEFXQGl/u0BE/3I/s4DX/7EXl7+ugGg/gA
D5wIC/hp+xgEFQOHF/vrRAeku/ttDQAEqrI/9ZAJlAQE01AAAAAEAAAAAA+oDKwACAAA3IQECA+j
+DMICaQAAAAACAAAAAAPqAxYAAgAFAAABIQkBIQEDYf0rAWr+DAPo/gwBFQGi/iACPwAAAAIAAAA
AA+oDuwAJABMAAAEzBxcnBzcnMzcHIQUDJQUDJSEDAj/kuUi8u0i55Eh0/oEBNXUBMwE0dQE1/oF
1AiWB04KC04HUltf+pdbWAVvXAVkAAAEAAAAAA+oD6gAJAAABBRMDJQUDEyUDAVz+pvo6ATUBNTz
6/qebAqE1/wD+lqurAWoBADUBSQAAAQAAAAAD6gMrAAIAACUBIQH2AfT8GMICaQAAAAIAAAAAA+o
DiAADAAsAADchEyE3FSMVITUhNUIDaT/8GKdPAyn+tmUCda5ERUVEAAUAAAAAA+ID6gAJABUALQA
5AFsAAAEeARcVITU+AT8BDgEHLgEnPgE3HgEFHgEfAQ4BBxUhNS4BJyM3PgE3LgEnDgEnDgEHLgE
nPgE3HgEFFBYfAQcOAQcVITUhNT4BNzMyFhc3Ji8BNz4BNy4BJw4BAsxcewL91wJ7XMUBTzo7TwE
BTzs6T/6uATQsAmaDAwKkAoNmAgIsNAECcVVVchwBPzAvPwICPy8wP/7kKSMDA1t1AgFy/swDgF8
fGDwaGxUWCAYgJQECYkpJYgE2A3pcHR1cegPnO08BAU87Ok8BAU86N1obARKXaltbapcSARtaNlV
yAgJxyy9AAQFALzA/AQE/MC1LGAEBHJJgdz06XHwDBgw3CgUCBBhIK0piAgJiAAADAAAAAAPqA+o
AFQAhAC0AAAEjDgEHHgEXPgE3NTM1IzUuAScjNSMBDgEHLgEnPgE3HgEFFgAXNgA3JgAnBgAB5AM
PEAEBHxURFwesrwIMCAMlAcgF97q59wUF97m69/xbBgEb09QBGwUF/uXU0/7lAiwIGhAWHwEBEAw
DKAMIDQTu/uC59wUF97m69wUF97rT/uUGBgEb09QBGwUF/uUAAgAAAAADugPpAAcAEgAANyERIxU
hNSMTFzM3ETMRFzM3ATADikv9DUyeAmuRTpJqAv7bAQFLpqYBeQKR/fUCC5ECASQAAAAAAv/+AAA
D6gMgADEAYwAAAQYHDgEHBiMGBw4BBwYXFhcWNyEWNzY3Njc1JyYnLgEnLgE1NicuAScmBwYnJic
mIyInBg8BJg4CFwcOARcWFxYzNhc3JyY3PgI3Mjc2Nz4BNzYXFh8BPgE3Jy4BJyYnIyIB+TAiGBo
```

```
FAQQhHRkiBwkNETscHQIWHhgeCAMBAQIFCycbBgMCERVAKiYiBAMNDy5AFhdXLQQtV1IiAhA9MAc
HLCUwJycGAyMPCCU5IAUBAwQLKR9IVD4oCREjEgIMPzUpKgYWAnoHIhg7IQUCDgsnGyopPRUKAQE
QFyUMDBEFERAaHwYCAwYkICYkAgERAwUQDyycFk4FFgI5UTIDEVQzNiIeAQEBBT1EHy4gBwYMDCI
4Ey0RDDELCAUBCzhSGBIBAAACAAAAAAPqA4sACQBFAAAlJwcXNycHESMRAy4BBw4BBw4BBx4BFzM
3Iy4BJz4BNzU3JjY3NhYXMDUzPgE3HgEXFR4BFAYHIxUzPgE3LgEnMS4BJw4BAfVJLZWULEk/Whx
JJyo2CkhYAgJ5X3wBekJZAgFSRwUCGCckOgpEA2BDRVcBSFZRR0RhSW8CAVZCA4BeR3LZSSyUlCx
JAVv+pQIrFhAKCjssE2pIVXQFPgJRPTxRAgIBG0EMCB0fAUdUAQNcOjgCVH9MAT4CclZIahlfeQI
BSwAAAAAFAAAAAAPqA+kACwAXADIAPgBKAAABFwcXNxc3JzcnBycFDgEHLgEnPgE3HgElBycmIyI
GBxc2PwEHDgEHHgEXPgE3LgEnIgYlFz4BMhYXNy4BIgYnFz4BIBYXNy4BIAYCOG5uLG5vLG9vLG9
uAUgCfFxdfAICfF1cfP6AAwYVFi5UIS0rPAcFFxkBBJ54d54DA553MFX+Xi06kqSSOi1Dqr2p7C1
c6QEE6VwtZv/+4v8Bim5uLG5uLG5vLG9vm117AgJ7XV17AwN7hAIBBSQhLCsGAQkiTyx3nwMDn3d
3nwMdUCs7Pz87K0VISGgrXmNjXitobGwAAAIAAAAAA+oD6gALAB8AAAEOAQcuASc+ATceAQUeARc
yNj8BATcBMT4BNy4BJw4BAn4Donl6ogMDonp5ov2HBMWVQXUuAwF3LP6IIycBBMWUlcUCjnqiAwO
ienmiAwOieZXFAy0oAv56LAGILW0+lMUDA8UABgAAAAADbQPqAAIABgAKAA0AFgAkAAAlNyc3Fzc
nNxc3JzcjNScVHgEXMxEhESMRHgEXIT4BNxEnISIGAR9sSCRIs0gkSEhIg3Y/ASgbl/2PPgEnGwJ
xGh8B6P49GyeiJEgkSLRHJEhISL9wLJ0bIgH9cANr/JUbIwEBIxsCvO0jAAAABgAAAAAD6wPpAAM
ABwAZACsALgBFAAAlMzUjNTM1IzcXFg4CIyEiLgI/AT4BMhYnBwYeAhchPgMvAS4BIgYTBSUnER4
BFyE1IREBFjI3AREzES4BJyEOAQKxPz8/P3GACwIYKhn+/xkpGQEKgQwrNivQgRECKkUqAQEqRSo
DEoEUSVlIr/69/r2CAREOAZn+hgF0CBQIAXQ/AREO/LYOEWE+PtsD+hUrKBcXKCsV+hgZGgb6JEt
FJgEBJkVLJPonKysBrOPjH/2wDhEBPwH1/voFBQEG/qMBmg0RAQERAAADAAAAAAPqAzAAGQAuADk
AAAEVFCsBIh0BFDsBMh0BFj8BNjQvATQnKwEGJREeATMhNTQ2OwE1NCcBBiInAQ4BNxcWMj8BNjc
hFBYDBAr6CQn6CgEF3gMD3gECAgH8/gEMCQGqDAqMCf7tCBMI/uoHBjL0BRAI8QUB/fICAmJ9Cgm
DCgl6CQbLAwcCzAIBAYv+mwkNgAoP3g8E/u0HBwEWAgst7gcH8QQIBAcAAwAAAAAD6gNtAAQABwA
LAAAlIREJAScJAQMhESEDrPyVAbUBtlT+nv6ekgPo/Bi+AmD+vAFFEP74AQj9UQLuAAcAAAAAA+o
CxwADAAgADAAQABMAFgAZAAATMzUjBSElByclMzUjNTM1IwUXEQE3JwUBIQLIyAESAs/+/mVm/ez
IyMjIAvfx/Sry8gFoAWf9MQGBIXycUVEsIUwhnZMBU/6tk8D3AR8ABAAAAAADzgPqAA8AIAAvADs
AACUjJzcjFwcjAwYCFyEuAQcBHgEXPgE3LgEnNDY1JiMOASUzMhYdARQGKwEVLwE3FQceARc+ATc
uAScOAQJeJjsvZDI2Lj+tjgMDryP1EP5EA5Rvb5MFY38CAxMTb5QCA3QKDAwKdzs4d5ACWUFDWAE
CWUFDWCuHV1SKARAm/v4Rv4sEASJxkwICkm8FgmAJEAcDA5OiDAooCgwpLChMIy9DVwICWUFDWAI
CWAAAAgAAAAAD3QPqAA0AGQAANxUhNS4BJw4BIiYnDgETHgEXPgE3LgEnDgEQA84DknEvc39zL3G
S1AOac3SaAwOadHOahIKCdaMRJCgoJBGjAeF0mgMDmnR0mgICmgAAABIA3gABAAAAAAAAAAEAAAA
BAAAAAAABAAoAAQABAAAAAAACAAcACwABAAAAAAADAAoAEgABAAAAAAAEAAoAHAABAAAAAAAFAAs
AJgABAAAAAAAGAAoAMQABAAAAAAAKACwAOwABAAAAAAALABIAZwADAAEECQAAAAIAeQADAAEECQA
BABQAewADAAEECQACAA4AjwADAAEECQADABQAnQADAAEECQAEABQAsQADAAEECQAFABYAxQADAAE
ECQAGABQA2wADAAEECQAKAFgA7wADAAEECQALACQBRyBlLXNiLWljb25zUmVndWxhcmUtc2ItaWN
vbnNlLXNiLWljb25zVmVyc2lvbiAxLjBlLXNiLWljb25zRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3l
uY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AcwBiAC0AaQBjAG8
AbgBzAFIAZQBnAHUAbABhAHIAZQAtAHMAYgAtAGkAYwBvAG4AcwBlAC0AcwBiAC0AaQBjAG8AbgB
zAFYAZQByAHMAaQBvAG4AIAAxAC4AMABlAC0AcwBiAC0AaQBjAG8AbgBzAEYAbwBuAHQAIABnAGU
AbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAB
yAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAA
CAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABoBAgEDAQQBBQEGAQcBCAEJAQoBCwEMAQ0
BDgEPARABEQESARMBFAEVARYBFwEYARkBGgEbAAltYWlsLXNlbmQLaW5ib3gtMDItd2YLaW5ib3g
tMDEtd2YFaW5ib3gPb3Blbi1tZXNzYWdlLXdmDGFycm93aGVhZC0wMRBhcnJvd2hlYWQtdG9wLXd
mCXJhdGluZy13ZgtyYXRpbmctLS0wMxFhcnJvd2hlYWQtZG93bi0wMQlmb2xkZXItMDMIdXNlcnM
td2YIY2xvY2stMDIGdXBsb2FkCG9uZWRyaXZlEWNsb3VkLWRvd25sb2FkLXdmD3dvcmstb2ZmbGl
uZS13ZglzZWFyY2gtd2YPbm90ZS1tZW1vLTAxLXdmDGltcG9ydGFudC13ZglzZW50LW1haWwIaW5
ib3gtd2YLbWFpbC0tLXNlbnQJdXNlci1iYWNrDHVzZXItcHJvZmlsZQAAAAA=)
format('truetype');
    font-weight: normal;
    font-style: normal;
  }

  [class^="sf-icon-"],
  [class*=" sf-icon-"] {
    font-family: 'e-sb-icons' ;
    speak: none;
```

```
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }

  span.e-text {
    color: #000000;
  }

  .title {
    text-align: center;
    font-size: 20px;
    padding: 15px;
  }

  .sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
  }

  .center {
    text-align: center;
    display: none;
    font-size: 13px;
    font-weight: 400;
    margin-top: 20px;
  }

  [class^="sf-icon-"],
  [class*=" sf-icon-"] {
    font-family: 'e-sb-icons' ;
    font-size: 15px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  body {
      margin-top:0px;
  }

  /* end of newTab support */
  li.e-list-item {
    border-bottom: 0.5px solid #80808070 !important;
    cursor: pointer;
  }
  .e-content-animation{
    height: 477px;
```

```
    margin-left:285px !important;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/fixed-position-cs1" %}

Sidebar with variation animation in Vue Sidebar component

In the following example, the sidebar is rendered with custom animation effects. Click the buttons available in the main content area to check how the custom animations works with sidebar.

Sidebar will automatically adjust expanding animation to match any custom size specified in CSS styles.

APP.VUE

```
<template>
    <div id="app">
       <div class="col-lg-8 control-section">
           <!-- Declaration of sidebar element-->
            <ejs-sidebar  id="sidebar-element"  class="sidebar"
:showBackdrop="showBackdrop" :closeOnDocumentClick="closeOnDocumentClick"
ref="sidebar" :width="width" >
                <div class="title"> Sidebar content</div>
                <div class="sub-title">
                    * Sidebar is rendered with animation effect
                </div>
                <div class="center-align">
                    <!-- Declaration of button element-->
                    <button  ejs-button  id="close_btn" v-
on:click="closeClick_btn"  class="e-btn close-btn">Close Sidebar</button>
                </div>
            </ejs-sidebar>
            <!-- End of sidebar element-->
        </div>
        <div class="content">
            <div class="title">Sidebar Transitions</div>
            <div class="sub-title"> * Click the below button to render the
Sidebar with animation effect.</div>
            <div class="center-align">
               <!-- Declaration of button element-->
                <button  ejs-button  id="zoom" v-on:click="zoom"  class="e-
btn e-info">Zoom Sidebar</button>
                <button  ejs-button  id="open_door" v-on:click="open_door"
class="e-btn e-info">Open Door</button>
                <button  ejs-button  id="bottom_top" v-on:click="bottom_top"
class="e-btn e-info">Bottom to Top</button>
            </div>
            <div class="center-align">
                <!-- Declaration of button element-->
                <button  ejs-button  id="rotate" v-on:click="rotate"
class="e-btn e-info">Rotate </button>
                <button  ejs-button  id="rotate_3d" v-on:click="rotate_3d"
class="e-btn e-info">Rotate 3D </button>
                <button  ejs-button  id="reverse" v-on:click="reverse"
class="e-btn e-info">Reverse Slide Out </button>
            </div>
        </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
</template>
<script>
import Vue from 'vue';
import { SidebarPlugin } from '@syncfusion/ej2-vue-navigations';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SidebarPlugin);
Vue.use(ButtonPlugin);
export default {
    data () {
        return {
            showBackdrop :true,
            closeOnDocumentClick:true,
            width: '220px',
        };
    },
    methods: {
        // Zoom Sidebar
        zoom :function() {
          this.$refs.sidebar.show();
          this.$refs.sidebar.$el.classList.add("w3-animate-zoom");
        },
        // Open Door
        open_door: function() {
            this.$refs.sidebar.show();
            const sidebar_element: Element
=document.getElementsByClassName("e-sidebar-overlay")[0];
            sidebar_element.classList.add("move");
        },
        // Bottom to Top
        bottom_top :function() {
          this.$refs.sidebar.show();
          this.$refs.sidebar.$el.classList.add("w3-animate-bottom");
        },
        // Rotate Sidebar
        rotate :function() {
          this.$refs.sidebar.show();
          this.$refs.sidebar.$el.classList.add("rotate");
        },
        // Rotate 3d Effect
        rotate_3d :function() {
          this.$refs.sidebar.show();
          this.$refs.sidebar.$el.classList.add("rotate_3d");
        },
        // Reverse Slide Out
        reverse :function() {
          this.$refs.sidebar.show();
          this.$refs.sidebar.$el.classList.add("reverse_slide_out");
        },
        //Close the sidebar
        closeClick_btn : function() {
            this.$refs.sidebar.$el.classList.remove("sidebar");
            this.$refs.sidebar.$el.classList.remove("rotate");
            this.$refs.sidebar.$el.classList.remove("w3-animate-zoom");
            this.$refs.sidebar.$el.classList.remove("w3-animate-bottom");
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            this.$refs.sidebar.$el.classList.remove("rotate_3d");
            this.$refs.sidebar.$el.classList.remove("reverse_slide_out");
            this.$refs.sidebar.hide();
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
/*Sidebar element styles*/
#sidebar-element {
    background-color: rgb(25, 118, 210);
    color: #ffffff;
}
.sidebar {
     animation-name:rotate_sidebar;
    animation-duration: 2s
}
@keyframes rotate_sidebar{from{transform:rotateY(150deg)}
to{transform:rotateY(360deg)}}
/*main content styles*/
.title {
    text-align: center;
    font-size: 25px;
    padding: 15px;
    font-style: italic;
}
.sub-title {
    text-align: center;
    font-size: 16px;
    padding: 10px;
    font-style: italic;
}
.center-align {
    text-align: center;
    padding-top: 20px;
}
/*Animation styles*/
.move {
    transform:rotateX(-20deg);
}
#close,#close:hover,#close:active,#close:focus{ /* csslint allow: adjoining-
classes*/
    background: #fafafa;
    color:black;
}
#close_btn,#close_btn:hover,#close_btn:active,#close_btn:focus{ /* csslint
allow: adjoining-classes*/
    background: #fafafa;
    color:black;
}
.w3-animate-bottom {
    animation-name: animatebottom;
    animation-duration: 1s;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@keyframes animatebottom{from {
    margin-top: 100%;
  }
  to {
    margin-top: 0%;
  }}

  .w3-animate-zoom {animation-name:animatezoom;
    animation-duration: 1s}
  @keyframes animatezoom{from{transform:scale(0)} to{transform:scale(1)}}
.rotate {
     animation-name:rotate1;
    animation-duration: 1s
}
@keyframes rotate1{from{transform:rotateX(150deg)}
to{transform:rotateX(360deg)}}
.rotate_3d {
    animation-name:rotate;
   animation-duration: 1s
}
@keyframes rotate{from{transform:rotateY(150deg)}
to{transform:rotateY(360deg)}}
  .reverse_slide_out  {
    animation-name: reverse1;
    animation-duration: 1s
 }
 @keyframes reverse1  {from {
    transform: rotateY(-65deg);
    margin-left: 200px;
  }
  to {
    margin-left: 0%;
  }
}
/*End of animation styles*/
button {
    margin:5px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sidebar/animation-cs1" %}

Signature

Getting Started with the Vue Signature Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Signature component using the Composition API / Options APILink to the Video.

To get start quickly with Vue Signature, you can check on this video:

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The list of dependencies required to use the Signature component in your application is given below:

`javascript

|-- @syncfusion/ej2-vue-inputs

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-inputs

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

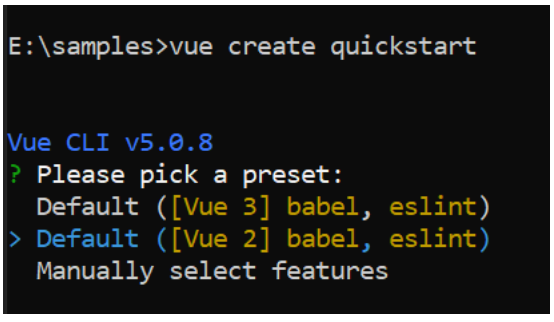
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Signature component as an example. Install the @syncfusion/ej2-vue-inputs package by running the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Add Syncfusion Vue component

Follow the below steps to add the Vue Signature component using Composition API or Options API:

1\. First, import and register the Signature component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SignatureComponent as EjsSignature } from '@syncfusion/ej2-vue-
inputs';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SignatureComponent } from '@syncfusion/ej2-vue-inputs';
export default {
components: {
'ejs-signature': SignatureComponent
}
}
</script>
```

2\. In the template section define the Signature component.

~/SRC/APP.VUE

```
<template>
<ejs-signature></ejs-signature>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class='wrap'>
    <h4>Sign here</h4>
```

```
    <ejs-signature id="signature"></ejs-signature>
</div>
</template>
<script setup>
import { SignatureComponent as EjsSignature } from '@syncfusion/ej2-vue-
inputs';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
#signature {
    border: 1px solid lightgray;
    height: 100%;
    width: 100%;
}
.wrap {
    height: 300px;
    width: 550px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class='wrap'>
    <h4>Sign here</h4>
    <ejs-signature id="signature"></ejs-signature>
</div>
</template>
<script>
import { SignatureComponent } from '@syncfusion/ej2-vue-inputs';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
    components: {
        'ejs-signature': SignatureComponent
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
#signature {
    border: 1px solid lightgray;
    height: 100%;
    width: 100%;
}
.wrap {
    height: 300px;
    width: 550px;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/signature/default-cs7" %}

Getting Started with the Vue Signature Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Signature component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

Copyright © 2001 -2024 Syncfusion Inc.

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Copyright © 2001 -2024 Syncfusion Inc.

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Signature component as an example. To use the Vue Signature component in the project, the @syncfusion/ej2-vue-inputs package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Signature component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue Signature component using Composition API or Options API:

1.First, import and register the Signature component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SignatureComponent as EjsSignature} from '@syncfusion/ej2-vue-
inputs';
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SignatureComponent } from '@syncfusion/ej2-vue-inputs';
//Component registration
export default {
name: "App",
components: {
'ejs-signature': SignatureComponent
}
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class='wrap'>
<h4>Sign here</h4>
<ejs-signature id="signature"></ejs-signature>
</div>
</template>
<script setup>
import { SignatureComponent as EjsSignature} from '@syncfusion/ej2-vue-
inputs';
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class='wrap'>
<h4>Sign here</h4>
<ejs-signature id="signature"></ejs-signature>
</div>
</template>
<script>
import { SignatureComponent } from '@syncfusion/ej2-vue-inputs';
// Component registration
export default {
name: "App",
// Declaring component and its directives
components: {
'ejs-signature': SignatureComponent
},
// Bound properties declarations
data() {
return {
};
}
};
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
#signature {
border: 1px solid lightgray;
height: 100%;
width: 100%;
}
.wrap {
height: 300px;
width: 550px;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

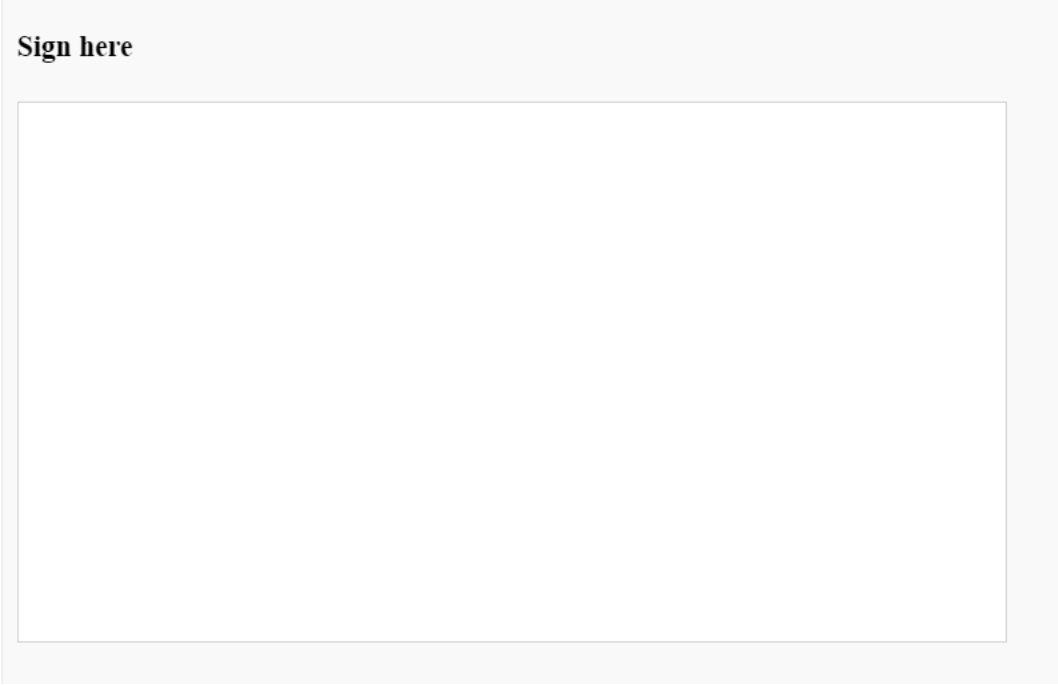
or

`bash

yarn run dev

`

The output will appear as follows:



For migrating from Vue 2 to Vue 3, refer to the migration documentation.

Copyright © 2001 -2024 Syncfusion Inc.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Customization in Vue Signature component

The Signature component draws stroke/path using moveTo() and lineTo() methods to connect one or more points while drawing in canvas. The stroke width can be modified by using its color and width. And the background can be modified by using its background color and background image.

Stroke Width

The variable stroke width is based on the values of maxStrokeWidth, minStrokeWidth and velocity for smoother and realistic signature. The default value of minimum stroke width is set as 0.5, maximum stroke width is set as 2.5 and velocity is set as 0.7.

In the following example, minimum stroke width is set as 0.5, maximum stroke width is set as 3 and velocity is set as 0.7.

APP.VUE

```
<template>
<div class='wrap'>
    <h4>Sign here</h4>
    <ejs-signature id="signature" :maxStrokeWidth="3" :mixStrokeWidth="0.5"
:velocity="0.7"></ejs-signature>
</div>
</template>
<script>
import Vue from 'vue';
import { SignaturePlugin } from '@syncfusion/ej2-vue-inputs';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SignaturePlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
#signature {
    border: 1px solid lightgray;
    height: 100%;
    width: 100%;
}
.wrap {
    height: 300px;
    width: 550px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/signature/default-cs2" %}

Stroke Color

Color of the stroke can be specified by using strokeColor property and it accepts hexadecimal code, RGB, and text. The default value of this property is “#000000”.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
<div class='wrap'>
    <div id="input">
        <input type="text" id="text" placeholder="Enter the Stroke Color
Value" >
        <ejs-button cssClass="e-primary" v-on:click.native="onSet">Set
Stroke Color</ejs-button>
    </div>
    <div id="signature-control">
        <ejs-signature id="signature"></ejs-signature>
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { SignaturePlugin } from "@syncfusion/ej2-vue-inputs";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { getComponent } from '@syncfusion/ej2-base';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SignaturePlugin);
Vue.use(ButtonPlugin);
export default {
   data: function() {
    return {
    };
  },
  methods: {
    onSet: function() {
        var signature = getComponent(document.getElementById('signature'),
'signature');
        var color = document.getElementById('text').value;
        signature.strokeColor = color;
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#signature {
  border: 1px solid lightgray;
  height: 100%;
  width: 100%;
}
#signature-control {
  height: 250px;
}
#input {
  margin-bottom: 30px;
}
#text {
  height: 30px;
  width: 300px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
.wrap {
    height: 300px;
    width: 550px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/signature/default-cs3" %}

Background Color

Background color of a signature can be specified by using backgroundColor property and it accepts hexadecimal code, RGB, and text. The default value of this property is “#ffffff”.

APP.VUE

```
<template>
<div class='wrap'>
    <div id="input">
        <input type="text" id="text" placeholder="Enter the Background Color Value" >
        <ejs-button cssClass="e-primary" v-on:click.native="onSet">Set Background Color</ejs-button>
    </div>
    <div id="signature-control">
        <ejs-signature id="signature"></ejs-signature>
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { SignaturePlugin } from "@syncfusion/ej2-vue-inputs";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { getComponent } from '@syncfusion/ej2-base';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SignaturePlugin);
Vue.use(ButtonPlugin);
export default {
   data: function() {
    return {
    };
  },
  methods: {
    onSet: function() {
        var signature = getComponent(document.getElementById('signature'), 'signature');
        var bgColor = document.getElementById('text').value;
        signature.backgroundColor = bgColor;
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
#signature {
  border: 1px solid lightgray;
  height: 100%;
  width: 100%;
}
#signature-control {
  height: 250px;
}
#input {
  margin-bottom: 30px;
}
#text {
  height: 30px;
  width: 300px;
}
.wrap {
    height: 300px;
    width: 550px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/signature/default-cs4" %}

Background Image

Background image of a signature can be specified by using backgroundImage property. The background image can be set by either hosting the image in our local IIS or online image.

APP.VUE

```
<template>
<div class='wrap'>
    <div id="input">
        <input type="text" id="text" placeholder="Enter the URL of the
background Image" >
        <ejs-button cssClass="e-primary" v-on:click.native="onSet">Set
Background Image</ejs-button>
    </div>
    <div id="signature-control">
        <ejs-signature id="signature"></ejs-signature>
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { SignaturePlugin } from "@syncfusion/ej2-vue-inputs";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { getComponent } from '@syncfusion/ej2-base';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SignaturePlugin);
Vue.use(ButtonPlugin);
export default {
   data: function() {
    return {
    };
  },
```

```
  methods: {
    onSet: function() {
        var signature = getComponent(document.getElementById('signature'),
'signature');
        var bgImage = document.getElementById('text').value;
        signature.backgroundImage = bgImage;
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#signature {
  border: 1px solid lightgray;
  height: 100%;
  width: 100%;
}
#signature-control {
  height: 250px;
}
#input {
  margin-bottom: 30px;
}
#text {
  height: 30px;
  width: 300px;
}
.wrap {
    height: 300px;
    width: 550px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/signature/default-cs5" %}

See Also

- Save With Background

Open save in Vue Signature component

The Signature component supports to open the signature by using hosted/online URL or base64. And it also supports various save options like image, base64, and blob.

Open Signature

The signature component opens a pre-drawn signature as either base64 or hosted/ online URL using the load method. It supports the PNG, JPEG, and SVG image's base64.

APP.VUE

```
<template>
<div class='wrap'>
    <div id="input">
```

```
        <input type="text" id="text" placeholder="Enter the Base64 or URL of
signature" >
        <ejs-button cssClass="e-primary" v-
on:click.native="onOpen">Open</ejs-button>
    </div>
    <div id="signature-control">
        <ejs-signature id="signature"></ejs-signature>
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { SignaturePlugin } from "@syncfusion/ej2-vue-inputs";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { getComponent } from '@syncfusion/ej2-base';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SignaturePlugin);
Vue.use(ButtonPlugin);
export default {
   data: function() {
    return {
    };
  },
  methods: {
    onOpen: function() {
        var signature = getComponent(document.getElementById('signature'),
'signature');
        var sign = document.getElementById('text').value;
        signature.load(sign);
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#signature {
  border: 1px solid lightgray;
  height: 100%;
  width: 100%;
}
#signature-control {
  height: 250px;
}
#input {
  margin-bottom: 30px;
}
#text {
  height: 30px;
  width: 300px;
}
.wrap {
    height: 300px;
    width: 550px;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/signature/default-cs9" %}

Save Signature

The Signature component saves the signature as base64, blob, and image like PNG, JPEG, and SVG.

Save as Base64

The getSignature method is used to get the signature as base64 with the PNG, JPEG, and SVG type. This can be loaded to signature using load method.

APP.VUE

```
<template>
<div class='wrap'>
    <span>Sign here</span>
    <div id="signature-control">
        <ejs-signature id="signature"></ejs-signature>
    </div>
    <ejs-button id="save" cssClass="e-primary" v-
on:click.native="onSave">Save as Base64</ejs-button>
    <ejs-dialog ref="dialogObj" header="Base64 of the signature"
:animationSettings="animationSettings" :showCloseIcon="true"
:visible="false" width="80%">
            </ejs-dialog>
</div>
</template>
<script>
import Vue from 'vue';
import { SignaturePlugin } from "@syncfusion/ej2-vue-inputs";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { getComponent } from '@syncfusion/ej2-base';
import { DialogPlugin } from '@syncfusion/ej2-vue-popups';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(DialogPlugin);
Vue.use(SignaturePlugin);
Vue.use(ButtonPlugin);
export default {
   data: function() {
    return {
      animationSettings: { effect: 'Zoom', duration: 400 },
    };
  },
  methods: {
    onSave: function(args) {
        var signature = getComponent(document.getElementById('signature'),
'signature');
        this.$refs.dialogObj.content = signature.getSignature();
        this.$refs.dialogObj.show();
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

```
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
#signature {
  border: 1px solid lightgray;
  height: 100%;
  width: 100%;
}
#signature-control {
  height: 300px;
  width: 550px;
}
#save {
  margin-top: 30px;
}
.wrap {
    height: 300px;
    width: 550px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/signature/default-cs10" %}

Save as Blob

The saveAsBlob method is used to saves the signature as Blob. It is defined as the chunk of binary data stored as a single entity in a database system.

Save as Image

The save method is used to saves the signature as an image. And it accepts file name and file type as parameter. The file type parameter supports PNG, JPEG, and SVG and the default file type is PNG.

APP.VUE

```
<template>
<div class='wrap'>
    <div>
        <span>Sign here</span>
        <ejs-splitbutton :items='items' content='Save' iconCss='e-sign-icons
e-save' :select="onSelect"></ejs-splitbutton>
    </div>
    <div id="signature-control">
        <ejs-signature id="signature"></ejs-signature>
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { SignaturePlugin } from "@syncfusion/ej2-vue-inputs";
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { getComponent } from '@syncfusion/ej2-base';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SignaturePlugin);
Vue.use(SplitButtonPlugin);
export default {
   data: function() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    return {
      items:[
            {
                text: 'Png'
            },
            {
                text: 'Jpeg'
            },
            {
                text: 'Svg'
            }]
    };
  },
  methods: {
    onSelect: function(args) {
        var signature = getComponent(document.getElementById('signature'),
'signature');
        signature.save(args.item.text, 'Signature');
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
#signature {
  border: 1px solid lightgray;
  height: 100%;
  width: 100%;
}
#signature-control {
  height: 300px;
  width: 550px;
}
.e-split-btn-wrapper {
  float: right;
  margin-bottom: 5px;
  margin-right: 50px;
}
@font-face {
  font-family: 'font-icons';
  src:
  url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tSfwAAAEoAAAAVmNtYXDOQM6IAAABqAAAAE5nbHlmPRF
AxQAAAhAAAAlsaGVhZB6WKa0AAADQAAAANmhoZWEIUQQLAAAArAAAACRobXR4KAAAAAAAAYAAAAA
obG9jYQowB4oAAAH4AAAAFm1heHABIAGEAAABCAAAACBuYW1lbLYTYgAAC3wAAAJJcG9zdIlCId8
AAA3IAAAAjwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAACgABAAAAAQAAc7rwy18
PPPUACwQAAAAAAN3B8l4AAAAA3cHyXgAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAKAXgADAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnCgQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAAAAAIAAAADAAAAFAADAAEAAAAUAAQAOgA
AAAYABAABAALnAecK//8AAOcA5wT//wAAAAAAAQAGAAgAAAABAAIAAwAEAAUABgAHAAgACQAAAAA
AAAA6AFoAiACyAOgCKAPQBFYEtgAAAAQAAAAA/QD8wADAAsAGQAjAAABESERARUzNTMVITUjESE
RMxUzESMRIREjESMRFSERIzUjNSEDHv3EAR5HSP6bSAH0j0dH/TZIRwPoR0j8pwFx/uIBHgI8j4/
```

```
X1/7iAR5I/O4BZv6aA1r8pkcDWUhHAAAAAwAAAAD7gP0AAMABwAPAAAlFSE1EzM1IwEhESMRIRE
jA0T9d1p8fP78A96r/XKl8WVlAgP//BkD6P7OATIAAAMAAAAAA/QD9AACAAYAGQAANyUnNxcBJzc
HFz8DNS8HDwIMASTqO+kB0+qpbulyBQQCAgQFpggJCQoJCQkMOuo66QHS6alu6XIICQoJCgkIpgc
EAwEBAwQAAAAABAAAAAAD9APqAAIABgAKAA8AACUHNyUBJwElByc3AQMlCQEBN8ctAj/+laMBbAF
PeaF6/XNQAVsCjf78nyzH+v6ZowFkC3ihd/3r/qxJAoABCwAAAgAAAAAD8wPoAB4AIgAAEw8HFR8
KMz8DFSE1IQE3CQI9BgsJBwcEAwICAwQHBwkLqgkJCQoJCQlGAo39iP7IPwE6AfH+xwGwBg0ODg8
PDxAQDxAPDw4ODaoGBAICBAZGM0gBOT7+xwHyATkAAgAAAAD8wPqAEkBGgAAAR8FDwwVHxM/CjU
vFCUzNT8RHxMVJx8BFQcfBh0BDw0rAS8OPwo1LwsjDwQBDwMVHxU7AT8DAT8EPQEvBTUvFg8TA4M
GBAMCAQEBAQQHHBAKCQcDAwECAQEDAwQFBgYHCAcJCAkICQkJCAkIBwgHBgYFBAMDAgECBQUHCQk
KDAwNDQ4ODw4dHBoiJv4aJgQCBAYGCAkLDA8ICAkJCgoLDAkKCQkJCAkIEA4ODQwLCQkHBgUEEwM
CAgcGBQUDAwIBAgIDBAQEBQUGBgYHBwcHBgcGBgYFBQUEAwMCAgEBAQICBAUFBgcHBQIDAgMDJgc
HBwcGBwYGCwsJBwv+oAMCAQEBBAUHChEVGRwVFhYWFxYgHxwTEBAODQUGBAUDAVwHBgUCAQIDBAU
DpAMEBggKCw0PCAkJCQkKCwsLCwwMDQ0NDQ0MCwsLCgkJCRAODAsJCAYFAwIB3AYFBgYGBgYGDQ0
tGhMVFwwMDQ4OCwsLCwoLCgoJCQgIBwYGBAQDAQEBAgMEBggJCwwOGfEVFRMSEhAQDg4NDAsKCgk
IDwsKCglwChgODg8ODw4NDAoFBAMDAwEBAQEBAgMDBQUFDRARExQWFxgYGBkYBhMdGBQPBQYGBgg
ICAkHBwcGBgYFBQQEBAMCAgEBAgIDBAQEBQUGBgYHBwcICAgHBwcGBRoaFBUXDA0NKAcFBAQCAgE
BAgQEB/6gBAQFBgYNDxASEh4gISEXFhYUExIYFBAIBwQCAgICBAFbCQsNBggHCAcICAgEoxgdHh4
eHh0cGgwMCwsKCQgIBwYFAwMCAQEBAgIDBAQFBgYMDxARERISEhIRAAAAAAUAAAAAA/QD5AA5AI4
AswDaAXcAAAEzHw8VDwcvBj0BLxUlHxMDDwUvFz8BHwk/BTUvDDclHwclLws1PwYfBicXDwQvCzU
/DTMfAycPDh8KDwQdAR8XOwE/CBMfAx0BHw07AT8NPQIvLCMPAQMxBwgODg0LCwoJCAcGBQQDAgE
BAQIDAwQEBAUEAwQCAgICBAMFCAQFBgUGBwcICQgKGxwcHh8V/sMGBg8SExUXFxkgIB8fHx4dHBs
SF+IDBQoJCgsMDg4QEBESExQUFRUWHBkiHRkUDwsHAwEBbiAaGxwdHR4eCAgIBwYGAgIBAgMFBSA
fHh0bGhgfWAEXBi0dIh8aFg/+1yAZHQ0LCQgHBQQDAgEDAwQJDhERExUXGBoc6QUJCQcGfxMPDg0
LCgkHBgUDAgEDBAMFBQYHCAgJDxAREhAQIyS5Dw4NDAsJCAcFAwMDAQEBAgMGCAoLDRcaExh0BAM
CAQICAwQJDA8RExUXFxoaGhIkJCMhIR8ODg4MCwsQDwkHBgYE1wMDAgEBAgMEBAUFBgcHBwgICAk
ICQgHCAcGBgYFBAMDAgICAgMFBQYGCAgKCgoMDBIJBwgKCwsNDQ4QDyMkJEMdHhwdHBwcGxoaGRg
XFxYWFBQCTAEDBAcICgsMDQ0PDxAQEBFtBAUDAwMCAQEBAQIDAwMFBDcsIAsWFgoKCgcHBgUFBAM
CBQQGCQsIqQoKExQTFBITERUSEA8NCwkGBQIF/ncEAwQBAQEBAQIDBAUGBwgJCgsMERAZGRcWFRI
QCgcGvxoWFBUTFBISAwEBAwUGBgYGBgYFBAQTFBQTFBUUG5gjAxkRGBgYFhIVFhUbDQ0LDAoKCgg
HBgUFAwIEAgEBAwQGBwoLWwEFBwgH3BUSEREQEBAODw0NDAwKCgkGBQUEBAQDAwMCAgEBBAZBBQU
HBwkJCgsICAgJCAkSExMTFBUUFR8gFRnJCAgICQkJCQkJChMUFBQTFBMSEhEQChMQDQwIBgIBAQI
EBgUFBgUGAXQTExYWNwgJCAgHBwYGBgUEAwMCAgICAwMEBQYGBgcHCAgJCG0RERAQEBAPDw4ODQw
LCgkKEwwNDQwNDQ0NDQ0ZGBUiDg4MCwsKCAgHBQUEAwEBAgMADAAAAAAD8gP0AAgADAAQABQAGAA
cAEQASABMAFAAVABYAAATFSE1JwcnBycFMzcjNxc3JwcXNyc/ATUnBxUXNRcVHwg/CD0BLwcrAQ8
HNxc3JwcXNyc7AScjJREhEQMhESF+AwSperIsRwFaCgYWRRwGFp8JHRCZIiLOIhkDBAYICgoGBgc
FDAoKCAYFAgEDBAYICQsGBgYGDAoKCAYFAgFxDxYGrBMPHEgWBgoBEPyuRAPk/BwBr96cVT+yGUs
DIhMWBxwcChYQLgcGBgYGChYJBgsLCQgHBQEBAQECBAcHCgsFBwYGCwsJCAcFAQECBQYICQsGBj8
QHAYGHw8WI1H9BQL7/GMD6AAAAAQAAAAAA/QDqAAGADYAPQBBAAABNxMVITUBJRUfCTsBPwk9AS8
KDwolEQMHAwERAyERIQJJg+v8kgEKAToBAQUHCAoGBQYHBgYGBgYGCQkHBAIBAQIEBwkJBgYGBgY
GBwYFBgoIBwQCAQEq7YL1/vY9A+j8GAFBqf7tQpYBR3oHBgYMCgkHAwICAQECAgMHCQoMBgYHBwY
GDAoJBwMCAQEBAQEBAgMHCQoMBgZ5/cgBF6gBMP64AeH87ANQAAAAAAAAEgDeAAEAAAAAAAAAAQA
AAAEAAAAAAAEACgABAAEAAAAAAAIABwALAAEAAAAAAAMACgASAAEAAAAAAAQACgAcAAEAAAAAAAU
ACwAmAAEAAAAAAAYACgAxAAEAAAAAAAoALAA7AAEAAAAAAAsAEgBnAAMAAQQJAAAAAgB5AAMAAQQ
JAAEAFAB7AAMAAQQJAAIADgCPAAMAAQQJAAMAFACdAAMAAQQJAAQAFACxAAMAAQQJAAUAFgDFAAM
AAQQJAAYAFADbAAMAAQQJAAoAWADvAAMAAQQJAAsAJAFHIGZvbnQtaWNvbnNSZWd1bGFyZm9udC1
pY29uc2ZvbnQtaWNvbnNWZXJzaW9uIDEuMGZvbnQtaWNvbnNGb250IGdlbmVyYXRlZCB1c2luZyB
TeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAGYAbwBuAHQALQBpAGM
AbwBuAHMAUgBlAGcAdQBsAGEAcgBmAG8AbgB0AC0AaQBjAG8AbgBzAGYAbwBuAHQALQBpAGMAbwB
uAHMAVgBlAHIAcwBpAG8AbgAgADEALgAwAGYAbwBuAHQALQBpAGMAbwBuAHMARgBvAG4AdAAgAGc
AZQBuAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB
0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMAbwBtAAA
AAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACgECAQMBBAEFAQYBBwEIAQkBCgELAAd
zYXZlXzAyB3NhdmUtMDEHZWRpdF8wMwdlZGl0XzAxBWNsZWFyDHBhaW50LWJ1Y2tldA9wYWludC1
idWNrZXQtd2YGaW1hZ2VzC3BpY3R1cmVzLXdmAAAA) format('truetype');
  font-weight: normal;
  font-style: normal;
}
```

```
.e-sign-icons {
  font-family: 'font-icons' !important;
  font-size: 55px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.e-save::before {
  content: '\e701';
}
.wrap {
    height: 300px;
    width: 550px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/signature/default-cs11" %}

Save with Background

The saveWithBackground property is used to saves the signature with its background and its default value is true. So, by default the signature is saved with its background.

In the following sample, the background color is set as 'rgb(103 58 183)' and save with background as true.

APP.VUE

```
<template>
<div class='wrap'>
    <div>
        <span>Sign here</span>
        <ejs-splitbutton :items='items' content='Save' iconCss='e-sign-icons
e-save' :select="onSelect"></ejs-splitbutton>
    </div>
    <div id="signature-control">
        <ejs-signature id="signature" backgroundColor="rgb(103 58 183)"
saveWithBackground="true"></ejs-signature>
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { SignaturePlugin } from "@syncfusion/ej2-vue-inputs";
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { getComponent } from '@syncfusion/ej2-base';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SignaturePlugin);
Vue.use(SplitButtonPlugin);
export default {
   data: function() {
    return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      items:[
            {
                text: 'Png'
            },
            {
                text: 'Jpeg'
            },
            {
                text: 'Svg'
            }]
    };
  },
  methods: {
    onSelect: function(args) {
        var signature = getComponent(document.getElementById('signature'),
'signature');
        signature.save(args.item.text, 'Signature');
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
#signature {
  border: 1px solid lightgray;
  height: 100%;
  width: 100%;
}
#signature-control {
  height: 300px;
  width: 550px;
}
.e-split-btn-wrapper {
  float: right;
  margin-bottom: 5px;
  margin-right: 50px;
}
@font-face {
  font-family: 'font-icons';
  src:
  url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tSfwAAAEoAAAAVmNtYXDOQM6IAAABqAAAAE5nbHlmPRF
AxQAAAhAAAAlsaGVhZB6WKa0AAADQAAAANmhoZWEIUQQLAAAArAAAACRobXR4KAAAAAAAAYAAAAA
obG9jYQowB4oAAAH4AAAAFm1heHABIAGEAAABCAAAACBuYW1lbLYTYgAAC3wAAAJJcG9zdIlCId8
AAA3IAAAAjwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAACgABAAAAAQAAc7rwy18
PPPUACwQAAAAAAN3B8l4AAAAA3cHyXgAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAKAXgADAAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnCgQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAAAAAIAAAADAAAAFAADAAEAAAAUAAQAOgA
AAAYABAABAALnAecK//8AAOcA5wT//wAAAAAAAQAGAAgAAAABAAIAAwAEAAUABgAHAAgACQAAAAA
AAAA6AFoAiACyAOgCKAPQBFYEtgAAAAQAAAAA/QD8wADAAsAGQAjAAABESERARUzNTMVITUjESE
RMxUzESMRIREjESMRFSERIzUjNSEDHv3EAR5HSP6bSAH0j0dH/TZIRwPoR0j8pwFx/uIBHgI8j4/
X1/7iAR5I/O4BZv6aA1r8pkcDWUhHAAAAAwAAAAD7gP0AAMABwAPAAAlFSE1EzM1IwEhESMRIRE
```

```
jA0T9d1p8fP78A96r/XKl8WVlAgP//BkD6P7OATIAAAMAAAAAA/QD9AACAAYAGQAANyUnNxcBJzc
HFz8DNS8HDwIMASTqO+kB0+qpbulyBQQCAgQFpggJCQoJCQkMOuo66QHS6alu6XIICQoJCgkIpgc
EAwEBAwQAAAAABAAAAAAD9APqAAIABgAKAA8AACUHNyUBJwElByc3AQMlCQEBN8ctAj/+laMBbAF
PeaF6/XNQAVsCjf78nyzH+v6ZowFkC3ihd/3r/qxJAoABCwAAAgAAAAAD8wPoAB4AIgAAEw8HFR8
KMz8DFSE1IQE3CQI9BgsJBwcEAwICAwQHBwkLqgkJCQoJCQlGAo39iP7IPwE6AfH+xwGwBg0ODg8
PDxAQDxAPDw4ODaoGBAICBAZGM0gBOT7+xwHyATkAAgAAAAAD8wPqAEkBGgAAAR8FDwwVHxM/CjU
vFCUzNT8RHxMVJx8BFQcfBh0BDw0rAS8OPwo1LwsjDwQBDwMVHxU7AT8DAT8EPQEvBTUvFg8TA4M
GBAMCAQEBAQQHHBAKCQcDAwECAQEDAwQFBgYHCAcJCAkICQkJCAkIBwgHBgYFBAMDAgECBQUHCQk
KDAwNDQ4ODw4dHBoiJv4aJgQCBAYGCAkLDA8ICAkJCgoLDAkKCQkJCAkIEA4ODQwLCQkHBgUEEwM
CAgcGBQUDAwIBAgIDBAQEBQUGBgYHBwcHBgcGBgYFBQUEAwMCAgEBAQICBAUFBgcHBQIDAgMDJgc
HBwcGBwYGCwsJBwv+oAMCAQEBBAUHChEVGRwVFhYWFxYgHxwTEBAODQUGBAUDAVwHBgUCAQIDBAU
DpAMEBggKCw0PCAkJCQkKCwsLCwwMDQ0NDQ0MCwsLCgkJCRAODAsJCAYFAwIB3AYFBgYGBgYGDQ0
tGhMVFwwMDQ4OCwsLCwoLCgoJCQgIBwYGBAQDAQEBAgMEBggJCwwOGfEVFRMSEhAQDg4NDAsKCgk
IDwsKCglwChgODg8ODw4NDAoFBAMDAwEBAQEBAgMDBQUFDRARExQWFxgYGBkYBhMdGBQPBQYGBgg
ICAkHBwcGBgYFBQQEBAMCAgEBAgIDBAQEBQUGBgYHBwcICAgHBwcGBRoaFBUXDA0NKAcFBAQCAgE
BAgQEB/6gBAQFBgYNDxASEh4gISEXFhYUExIYFBAIBwQCAgICBAFbCQsNBggHCAcICAgEoxgdHh4
eHh0cGgwMCwsKCQgIBwYFAwMCAQEBAgIDBAQFBgYMDxARERISEhIRAAAAAAAUAAAAAA/QD5AA5AI4
AswDaAXcAAAEzHw8VDwcvBj0BLxUlHxMDDwUvFz8BHwk/BTUvDDclHwclLws1PwYfBicXDwQvCzU
/DTMfAycPDh8KDwQdAR8XOwE/CBMfAx0BHw07AT8NPQIvLCMPAQMxBwgODg0LCwoJCAcGBQQDAgE
BAQIDAwQEBAUEAwQCAgICBAMFCAQFBgUGBwcICQgKGxwcHh8V/sMGBg8SExUXFxkgIB8fHx4dHBs
SF+IDBQoJCgsMDg4QEBESExQUFRUWHBkiHRkUDwsHAwEBbiAaGxwdHR4eCAgIBwYGAgIBAgMFBSA
fHh0bGhgfWAEXBi0dIh8aFg/+1yAZHQ0LCQgHBQQDAgEDAwQJDhEREXUXGBoc6QUJCQcGfxMPDg0
LCgkHBgUDAgEDBAMFBQYHCAgJDxAREhAQIyS5Dw4NDAsJCAcFAwMDAQEBAgMGCAoLDRcaExh0BAM
CAQICAwQJDA8REXUXFxoaGhIkJCMhIR8ODg4MCwsQDwkHBgYE1wMDAgEBAgMEBAUFBgcHBwgICAk
ICQgHCAcGBgYFBAMDAgICAgMFBQYGCAgKCgoMDBIJBwgKCwsNDQ4QDyMkJEMdHhwdHBwcGxoaGRg
XFxYWFBQCTAEDBAcICgsMDQ0PDxAQEBFtBAUDAwMCAQEBAQIDAwMFBDcsIAsWFgoKCgcHBgUFBAM
CBQQGCQsIqQoKExQTFBITERUSEA8NCwkGBQIF/ncEAwQBAQEBAQIDBAUGBwgJCgsMERAZGRcWFRI
QCgcGvxoWFBUTFBISAwEBAwUGBgYGBgYFBAQTFBQTFBUUG5gjAxkRGBgYFhIVFhUbDQ0LDAoKCgg
HBgUFAwIEAgEBAwQGBwoLWwEFBwgH3BUSEREQEBAODw0NDAwKCgkGBQUEBAQDAwMCAgEBBAZBBQU
HBwkJCgsICAgJCAkSExMTFBUUFR8gFRnJCAgICQkJCQkJChMUFBQTFBMSEhEQChMQDQwIBgIBAQI
EBgUFBgUGAXQTExYWNwgJCAgHBwYGBgUEAwMCAgICAwMEBQYGBgcHCAgJCG0RERAQEBAPDw4ODQw
LCgkKEwwNDQwNDQ0NDQ0ZGBUiDg4MCwsKCAgHBQUEAwEBAgMADAAAAAAD8gP0AAgADAAQABQAGAA
cAEQASABMAFAAVABYAAATFSE1JwcnBycFMzcjNxc3JwcXNyc/ATUnBxUXNRcVHwg/CD0BLwcrAQ8
HNxc3JwcXNyc7AScjJREhEQMhESF+AwSperIsRwFaCgYWRRwGFp8JHRCZIiLOIhkDBAYICgoGBgc
FDAoKCAYFAgEDBAYICQsGBgYGDAoKCAYFAgFxDxYGrBMPHEgWBgoBEPyuRAPk/BwBr96cVT+yGUs
DIhMWBxwcChYQLgcGBgYGChYJBgsLCQgHBQEBAQECBAcHCgsFBwYGCwsJCAcFAQECBQYICQsGBj8
QHAYGHw8WI1H9BQL7/GMD6AAAAAQAAAAAA/QDqAAGADYAPQBBAAABNxMVITUBJRUfCTsBPwk9AS8
KDwolEQMHAwERAyERIQJJg+v8kgEKAToBAQUHCAoGBQYHBgYGBgYGCQkHBAIBAQIEBwkJBgYGBgY
GBwYFBgoIBwQCAQEq7YL1/vY9A+j8GAFBqf7tQpYBR3oHBgYMCgkHAwICAQECAgMHCQoMBgYHBwY
GDAoJBwMCAQEBAQEBAgMHCQoMBgZ5/cgBF6gBMP64AeH87ANQAAAAAAAAEgDeAAEAAAAAAAAAAAQA
AAAEAAAAAAAEACgABAAEAAAAAAAIABwALAAEAAAAAAAMACgASAAEAAAAAAAQACgAcAAEAAAAAAAU
ACwAmAAEAAAAAAAYACgAxAAEAAAAAAAoALAA7AAEAAAAAAAsAEgBnAAMAAQQJAAAAAgB5AAMAAQQ
JAAEAFAB7AAMAAQQJAAIADgCPAAMAAQQJAAMAFACdAAMAAQQJAAQAFACxAAMAAQQJAAUAFgDFAAM
AAQQJAAYAFADbAAMAAQQJAAoAWADvAAMAAQQJAAsAJAFHIGZvbnQtaWNvbnNSZWd1bGFyZm9udC1
pY29uc2ZvbnQtaWNvbnNWZXJzaW9uIDEuMGZvbnQtaWNvbnNGb250IGdlbmVyYXRlZCB1c2luZyB
TeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAGYAbwBuAHQALQBpAGM
AbwBuAHMAUgBlAGcAdQBsAGEAcgBmAG8AbgB0AC0AaQBjAG8AbgBzAGYAbwBuAHQALQBpAGMAbwB
uAHMAVgBlAHIAcwBpAG8AbgAgADEALgAwAGYAbwBuAHQALQBpAGMAbwBuAHMARgBvAG4AdAAgAGc
AZQBuAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB
0AHIAbwAgAFMAdAB1AGQaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMAbwBtAAA
AAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACgECAQMBBAEFAQYBBwEIAQkBCgELAAd
zYXZlXzAyB3NhdmUtMDEHZWRpdF8wMwdlZGl0XzAxBWNsZWFyDHBhaW50LWJ1Y2tldA9wYWludC1
idWNrZXQtd2YGaW1hZ2VzC3BpY3R1cmVzLXdmAAAA) format('truetype');
  font-weight: normal;
  font-style: normal;
}
.e-sign-icons {
```

```
  font-family: 'font-icons' !important;
  font-size: 55px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.e-save::before {
  content: '\e701';
}
.wrap {
    height: 300px;
    width: 550px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/signature/default-cs12" %}

Draw in Vue Signature component

Draw

The draw method is used to drawn a text as signature with different font families like Arial, Serif, with different font sizes. It accepts text, font family, font size as its parameters. The default font family is “Arial”, and the default font size is “30”.

APP.VUE

```
<template>
<div class='wrap'>
    <div id="signature-control">
            <ejs-signature id="signature"></ejs-signature>
        </div>
        <div id="input">
            <table>
                <tbody>
                    <tr>
                        <td><div>Enter the Text:</div></td>
                        <td><input type="text" id="text" placeholder="Enter
the Text"></td>
                    </tr>
                    <tr>
                        <td style="padding-top:10px"><div>Font
Family:</div></td>
                        <td style="padding-top:10px">
                          <ejs-dropdownlist ref="font"
:dataSource='fontsData' :value="fontValue" @change="fontChange"></ejs-
dropdownlist>
                        </td>
                    </tr>
                    <tr>
                        <td style="padding-top:10px"><div>Font
Size:</div></td>
                        <td style="padding-top:10px">
```

```
                            <ejs-dropdownlist ref="size"
:dataSource='sizeData' :value="sizeValue"  @change="sizeChange"></ejs-
dropdownlist>
                        </td>
                    </tr>
                </tbody>
            </table>
            <br>
            <ejs-button cssClass="e-primary" v-
on:click.native="onDraw">Draw</ejs-button>
        </div>
</div>
</template>
<script>
import Vue from 'vue';
import { SignaturePlugin } from "@syncfusion/ej2-vue-inputs";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { getComponent } from '@syncfusion/ej2-base';
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(DropDownListPlugin);
Vue.use(SignaturePlugin);
Vue.use(ButtonPlugin);
export default {
   data: function() {
    return {
      fontValue: "Arial", sizeValue: 20,
      fontsData: ['Arial', 'Serif', 'Sans-serif', 'Cursive', 'Fantasy'],
      sizeData: [ 20, 30, 40, 50],
    };
  },
  methods: {
    onDraw: function(args) {
        var signature = getComponent(document.getElementById('signature'),
'signature');
        var text = document.getElementById('text').value;
        signature.draw(text, this.$refs.font.value, this.$refs.size.value);
    },
    fontChange: function(args) {
      this.fontValue = args.value;
    },
    sizeChange: function(args) {
      this.sizeValue = args.value;
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
#signature {
  border: 1px solid lightgray;
  height: 100%;
  width: 100%;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
#signature-control {
  height: 300px;
  width: 500px;
  float: left;
}
#input {
  margin-top: 30px;
  margin-right: 20px;
  float: right;
}
#text {
  height: 30px;
}
.wrap {
    height: 300px;
    width: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/signature/default-cs6" %}

User interaction in Vue Signature component

The below interactions were available in Signature, and we can walk through one by one.

- Undo and Redo
- Clear
- Disabled
- ReadOnly

Undo and Redo

In the Signature, every action can be maintained as a snap for undo and redo operations. And maintained SnapIndex for indexing the snap collection.

The undo method reverts the last action of signature by decreasing SnapIndex value to index previous snap. Here, canUndo method is used to ensure whether undo can be performed or not.

The redo method reverts the last undo action of the signature by increasing the SnapIndex to get the next snap. Here, canRedo method is used to ensure whether redo can be performed or not.

Clear

The clear method is used to clears the signature and makes the canvas empty. This is also considered in Undo/ Redo. Here, isEmpty method is used to ensure whether the signature is empty or not.

Disabled

The disabled property is used to enables/disables the signature component. In the disabled state, the user is not allowed to draw signature. And it can’t be focused until the user enabled the signature.

ReadOnly

The isReadOnly property is used to enables/disables the ReadOnly Signature. It can be focused but it prevents drawing in Signature.

The following sample explains about user interactions available in signature.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
<div class='wrap'>
    <div id="option">
            <table>
                <tr>
                    <td>
                       <ejs-button id="undo" cssClass="e-primary"
:disabled="undoDisable" v-on:click.native="onUndo">UNDO</ejs-button>
                    </td><td>
                        <ejs-button id="redo" cssClass="e-primary"
:disabled="redoDisable" v-on:click.native="onRedo">REDO</ejs-button>
                    </td><td>
                        <ejs-button id="clear" cssClass="e-primary"
:disabled="clearDisable" v-on:click.native="onClear">CLEAR</ejs-button>
                    </td><td>
                        <div style="margin-bottom: 5px; margin-left:
200px;"><ejs-checkbox label='Disabled' :change="disableChange"></ejs-
checkbox></div>
                        <div style="margin-left: 200px;"><ejs-checkbox
label='ReadOnly' :change="readOnlyChange"></ejs-checkbox></div>
                    </td>
                </tr>
            </table>
        </div>
        <div id ="signature-control">
            <ejs-signature id="signature" :change="onChange"></ejs-
signature>
        </div>
</div>
</template>
<script>
import Vue from 'vue';
import { SignaturePlugin } from "@syncfusion/ej2-vue-inputs";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { getComponent } from '@syncfusion/ej2-base';
import { CheckBoxPlugin } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(CheckBoxPlugin);
Vue.use(SignaturePlugin);
Vue.use(ButtonPlugin);
export default {
   data: function() {
    return {
      undoDisable: true, redoDisable: true, clearDisable: true,
    };
  },
  methods: {
    onUndo: function(args) {
        var signature = getComponent(document.getElementById('signature'),
'signature');
        if (!signature.disabled && !signature.isReadOnly) {
          signature.undo();
        }
    },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    onRedo: function(args) {
      var signature = getComponent(document.getElementById('signature'),
'signature');
        if (!signature.disabled && !signature.isReadOnly) {
          signature.redo();
        }
    },
    onClear: function(args) {
      var signature = getComponent(document.getElementById('signature'),
'signature');
        if (!signature.disabled && !signature.isReadOnly) {
          signature.clear();
        }
    },
    disableChange: function(args) {
      var signature = getComponent(document.getElementById('signature'),
'signature');
      signature.disabled = args.checked;
    },
    readOnlyChange: function(args) {
      var signature = getComponent(document.getElementById('signature'),
'signature');
      signature.isReadOnly = args.checked;
    },
    onChange: function() {
      var signature = getComponent(document.getElementById('signature'),
'signature');
      if (!signature.disabled && !signature.isReadOnly) {
        if (signature.canUndo()) {
            this.undoDisable = false;
        } else {
            this.undoDisable = true;
        }
        if (signature.canRedo()) {
            this.redoDisable = false;
        } else {
            this.redoDisable = true;
        }
        if (!signature.isEmpty()) {
            this.clearDisable = false;
        } else {
            this.clearDisable = true;
        }
    }
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
#signature {
  border: 1px solid lightgray;
  height: 100%;
  width: 100%;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
#signature-control {
  height: 300px;
  width: 550px;
}
#option {
  margin-bottom: 10px;
}
.wrap {
    height: 300px;
    width: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/signature/default-cs13" %}

Accessibility in Vue Signature component

The Signature component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Signature component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | Not Applicable |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

Copyright © 2001 -2024 Syncfusion Inc.

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

Keyboard interaction

The Signature component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Signature component.

| **Press** | **To do this** |

| --- | --- |

| Ctrl + Z | Undo the last action. |

| Ctrl + Y | Redo the last action. |

| Ctrl + S | To save the signature. |

| delete | Erases all the signature strokes signed by user. |

Ensuring accessibility

The Signature component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Signature component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Signature component with accessibility tools.

{% previewsample "page.domainurl/code-snippet/signature/default-cs1" %}

See also

- Accessibility in Syncfusion Vue components

How To

Integrate toolbar in Vue Signature component

The Signature component integrates with the toolbar and the interaction performed using the change event of the toolbar. In that, canUndo, canRedo and isEmpty methods were used to enable/disable undo, redo, and clear buttons.

APP.VUE

```
<template>
  <div>
    <div class="col-lg-12 control-section">
```

```
      <div id="signature-toolbar-control">
        <ejs-toolbar id="toolbar" :created="onCreate" :clicked="clicked">
          <e-items>
            <e-item text='Undo' prefixIcon='e-icons e-undo'
tooltipText='Undo (Ctrl + Z)'></e-item>
            <e-item text='Redo' prefixIcon='e-icons e-redo'
tooltipText='Redo (Ctrl + Y)' ></e-item>
            <e-item type='Separator'></e-item>
            <e-item tooltipText= 'Save (Ctrl + S)' :template=
'SaveTemplate'></e-item>
            <e-item type='Separator'></e-item>
            <e-item tooltipText= 'Stroke Color' :template=
'StrokeColorTemplate'></e-item>
            <e-item type='Separator'></e-item>
            <e-item tooltipText= 'Background Color' :template=
'BgColorTemplate'></e-item>
            <e-item type='Separator'></e-item>
            <e-item tooltipText= 'Stroke Width' :template=
'StrokeWidthTemplate'></e-item>
            <e-item type='Separator'></e-item>
            <e-item  text= 'Clear' prefixIcon= 'e-sign-icons e-clear'
tooltipText= 'Clear'></e-item>
            <e-item tooltipText= 'Disabled' :template='DisableTemplate'
align='Right'></e-item>
          </e-items>
        </ejs-toolbar>
        <div id="signature-control">
          <ejs-signature #signature id="signature"
:maxStrokeWidth="maxStrokeWidth" style="height: 90%; width:100%;"
:change="signChange"></ejs-signature>
        </div>
      </div>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { SignaturePlugin, ColorPickerPlugin } from "@syncfusion/ej2-vue-
inputs";
import { ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { CheckBoxPlugin, ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { getComponent, addClass, createElement } from '@syncfusion/ej2-
base';
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SignaturePlugin);
Vue.use(ColorPickerPlugin);
Vue.use(ToolbarPlugin);
Vue.use(SplitButtonPlugin);
Vue.use(CheckBoxPlugin);
Vue.use(ButtonPlugin);
Vue.use(DropDownListPlugin);
export default {
  data: function() {
    return {
```

```
      maxStrokeWidth: 2,
      SaveTemplate: function () {
        return {
            template: Vue.component('SplitButtonComponent', {
            template: '<ejs-splitbutton content="Save" id="save-option"
:items="items" iconCss= "e-sign-icons e-save" :select="onSelect"
disabled="true"></ejs-splitbutton>',
            data() { return {
                items: [
                {
                  text: 'Png'
                },
                {
                  text: 'Jpeg'
                },
                {
                  text: 'Svg'
                }
              ]};
            },
            methods: {
              onSelect: function(args) {
                var signature =
getComponent(document.getElementById('signature'), 'signature');
                signature.save(args.item.text, 'Signature');
             }
            }
          })
        }
      },
      StrokeColorTemplate: function () {
        return {
            template: Vue.component('ColorPickerComponent', {
            template: '<ejs-colorpicker id="stroke-color" mode="Palette"
cssClass="e-stroke-color" :change="strokeColorChanged"  :modeSwitcher=
"false" :columns="4" :presetColors="customColors"
:beforeTileRender="beforeTileRender" :showButtons= "false"></ejs-
colorpicker>',
            data() { return {
              customColors:{
                'custom': ['#000000', '#e91e63', '#9c27b0', '#673ab7',
'#2196f3', '#03a9f4', '#00bcd4',
                '#009688', '#8bc34a', '#cddc39', '#ffeb3b', '#ffc107']
              }
            };},
            methods: {
              beforeTileRender: function(args) {
                args.element.classList.add('e-circle-palette');
                args.element.appendChild(createElement('span', { className:
'e-circle-selection' }));
              },
              strokeColorChanged: function(args) {
                var signature =
getComponent(document.getElementById('signature'), 'signature');
                var strokecolor =
getComponent(document.getElementById('stroke-color'), 'colorpicker');
                if (signature.disabled) {
```

```
                    return;
                  }
                  var selElem=
strokecolor.element.nextElementSibling.querySelector('.e-selected-color');
                  selElem.style.borderBottomColor = args.currentValue.rgba;
                  signature.strokeColor = args.currentValue.rgba;
                }
              }
            })
          }
        },
        BgColorTemplate: function () {
          return {
            template: Vue.component('ColorPickerComponent', {
              template: '<ejs-colorpicker id="bg-color" noColor="true"
mode="Palette" cssClass="e-bg-color" :change="bgColorChanged" :modeSwitcher=
"false" :columns="4" :presetColors="customColors"
:beforeTileRender="beforeTileRender1" :showButtons= "false"></ejs-
colorpicker>',
              data() { return {
                customColors:{
                  'custom': ['#ffffff', '#f44336', '#e91e63', '#9c27b0',
'#673ab7', '#2196f3', '#03a9f4', '#00bcd4',
                  '#009688', '#8bc34a', '#cddc39', '#ffeb3b']
                }
              };},
              methods: {
                beforeTileRender1: function(args) {
                  args.element.classList.add('e-circle-palette');
                  args.element.appendChild(createElement('span', { className:
'e-circle-selection' }));
                },
                bgColorChanged: function(args) {
                  var signature =
getComponent(document.getElementById('signature'), 'signature');
                  var bgcolor = getComponent(document.getElementById('bg-
color'), 'colorpicker');
                  if (signature.disabled) {
                    return;
                  }
                  var selElem=
bgcolor.element.nextElementSibling.querySelector('.e-selected-color');
                  signature.backgroundColor = args.currentValue.rgba;
                  selElem.style.borderBottomColor = args.currentValue.rgba;
                }
              }
            })
          }
        },
        DisableTemplate: function () {
          return {
            template: Vue.component('CheckBoxComponent', {
              template: '<ejs-checkbox label="Disabled"
:change="change1"></ejs-checkbox>',
              data() { return {
              };},
              methods: {
```

```
                change1: function(args) {
                  var signature =
getComponent(document.getElementById('signature'), 'signature');
                  signature.disabled = args.checked;
                }
              }
            })
          }
        },
        StrokeWidthTemplate: function () {
          return {
            template: Vue.component('DropDownListComponent', {
              template: '<ejs-dropdownlist  :value="value" width="60"
:dataSource="items":change="numChange"></ejs-dropdownlist>',
              data() { return {
                value: 2,
                items:[1, 2, 3, 4, 5]
              };},
              methods: {
                numChange: function(args) {
                  var signature =
getComponent(document.getElementById('signature'), 'signature');
                  signature.maxStrokeWidth = args.value;
                }
              }
            })
          }
        }
      };
    },
    methods: {
      onCreate: function() {
        var strokeColor = getComponent(document.getElementById('stroke-
color'), 'colorpicker');
        var bgColor = getComponent(document.getElementById('bg-color'),
'colorpicker');
        addClass([strokeColor.element.nextElementSibling.querySelector('.e-
selected-color')], 'e-sign-icons');
        addClass([bgColor.element.nextElementSibling.querySelector('.e-
selected-color')], 'e-sign-icons');
        document.getElementById('save-option').addEventListener('click',
this.saveBtnClick);
        var cBtn = this.clearButton();
        cBtn.disabled = true;
        var toolbarlItems = document.querySelectorAll('.e-toolbar .e-toolbar-
items .e-toolbar-item .e-tbar-btn.e-tbtn-txt');
        for (var i = 0; i < toolbarlItems.length; i++) {
          if (toolbarlItems[i].children[0].classList.contains('e-undo')) {
            var undoButton = getComponent(toolbarlItems[i] , 'btn');
            undoButton.disabled = true;
          }
          if (toolbarlItems[i].children[0].classList.contains('e-redo')) {
            var redoButton= getComponent(toolbarlItems[i], 'btn');
            redoButton.disabled = true;
          }
        }
      },
```

```
    clicked:function(args) {
      var signature = getComponent(document.getElementById('signature'),
'signature');
      var saveBtn =  getComponent(document.getElementById("save-option"),
'split-btn');
      if (signature.disabled) {
        return;
      }
      switch (args.item.tooltipText) {
        case 'Undo (Ctrl + Z)':
          if (signature.canUndo()) {
            signature.undo();
            this.updateUndoRedo();
            this.updateSaveBtn();
          }
          break;
        case 'Redo (Ctrl + Y)':
          if (signature.canRedo()) {
            signature.redo();
            this.updateUndoRedo();
            this.updateSaveBtn();
          }
          break;
        case 'Clear':
          signature.clear();
          if (signature.isEmpty()) {
            var cBtn = this.clearButton();
            cBtn.disabled = true;
            saveBtn.disabled = true;
          }
          break;
        case 'Disabled':
          signature.disabled = !(args.item.template).checked;
      }
    },
    signChange: function() {
      var signature = getComponent(document.getElementById('signature'),
'signature');
      var saveBtn =  getComponent(document.getElementById("save-option"),
'split-btn');
      if (!signature.isEmpty()) {
        var cBtn = this.clearButton();
        saveBtn.disabled = false;
        cBtn.disabled = false;
      }
      this.updateUndoRedo();
    },
    saveBtnClick: function() {
      var signature = getComponent(document.getElementById('signature'),
'signature');
      signature.save();
    },
    clearButton: function() {
      var tlItems = document.querySelectorAll('.e-toolbar .e-toolbar-items
.e-toolbar-item .e-tbar-btn.e-tbtn-txt');
      for (var i = 0; i < tlItems.length; i++) {
        if (tlItems[i].children[0].classList.contains('e-clear')) {
```

```
            var clrBtn = getComponent(tlItems[i], 'btn');
          }
        }
        return clrBtn;
      },
      updateSaveBtn: function() {
          var signature = getComponent(document.getElementById('signature'),
'signature');
          var saveBtn = getComponent(document.getElementById("save-option"),
'split-btn');
          if (signature.isEmpty()) {
              saveBtn.disabled = true;
          }
      },
      updateUndoRedo: function() {
        var signature = getComponent(document.getElementById('signature'),
'signature');
        var undoButton; var redoButton;
        var tlItems = document.querySelectorAll('.e-toolbar .e-toolbar-items
.e-toolbar-item .e-tbar-btn.e-tbtn-txt');
        for (var i = 0; i < tlItems.length; i++) {
          if (tlItems[i].children[0].classList.contains('e-undo')) {
            undoButton = getComponent(tlItems[i], 'btn');
          }
          if (tlItems[i].children[0].classList.contains('e-redo')) {
            redoButton = getComponent(tlItems[i], 'btn');
          }
        }
        if (signature.canUndo()) {
          undoButton.disabled = false;
        } else {
          undoButton.disabled = true;
        }
        if (signature.canRedo()) {
          redoButton.disabled = false;
        } else {
          redoButton.disabled = true;
        }
      }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-lists/styles/material.css";
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-dropdowns/styles/material.css";
@font-face {
  font-family: 'font-icons';
  src:
  url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tSfwAAAEoAAAAVmNtYXDOQM6IAAABqAAAAE5nbHlmPRF
AxQAAAhAAAAlsaGVhZB6WKa0AAADQAAAANmhoZWEIUQQLAAAArAAAACRobXR4KAAAAAAAAYAAAAA
```

```
obG9jYQowB4oAAAH4AAAAFm1heHABIAGEAAABCAAAACBuYW1lbLYTYgAAC3wAAAJJcG9zdIlCId8
AAA3IAAAAjwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAACgABAAAAAQAAc7rwy18
PPPUACwQAAAAAAN3B8l4AAAAA3cHyXgAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAKAXgADAAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnCgQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAAAAAIAAAADAAAAFAADAAEAAAAUAAQAOgA
AAAYABAABAALnAecK//8AAOcA5wT//wAAAAAAAQAGAAgAAAABAAIAAwAEAAUABgAHAAgACQAAAAA
AAAA6AFoAiACyAOgCKAPQBFYEtgAAAAQAAAAAA/QD8wADAAsAGQAjAAABESERARUzNTMVITUjESE
RMxUzESMRIREjESMRFSERIzUjNSEDHv3EAR5HSP6bSAH0j0dH/TZIRwPoR0j8pwFx/uIBHgI8j4/
X1/7iAR5I/O4BZv6aA1r8pkcDWUhHAAAAAwAAAAD7gP0AAMABwAPAAAlFSE1EzM1IwEhESMRIRE
jA0T9d1p8fP78A96r/XKl8WVlAgP//BkD6P7OATIAAAMAAAAAA/QD9AACAAYAGQAANyUnNxcBJzc
HFz8DNS8HDwIMASTqO+kB0+qpbulyBQQCAgQFpggJCQoJCQkMOuo66QHS6alu6XIICQoJCgkIpgc
EAwEBAwQAAAAABAAAAAAD9APqAAIABgAKAA8AACUHNyUBJwElByc3AQMlCQEBN8ctAj/+laMBbAF
PeaF6/XNQAVsCjf78nyzH+v6ZowFkC3ihd/3r/qxJAoABCwAAAgAAAAD8wPoAB4AIgAAEw8HFR8
KMz8DFSE1IQE3CQI9BgsJBwcEAwICAwQHBwkLqgkJCQoJCQlGAo39iP7IPwE6AfH+xwGwBg0ODg8
PDxAQDxAPDw4ODaoGBAICBAZGM0gBOT7+xwHyATkAAgAAAAD8wPqAEkBGgAAAR8FDwwVHxM/CjU
vFCUzNT8RHxMVJx8BFQcfBh0BDw0rAS8OPwo1LwsjDwQBDwMVHxU7AT8DAT8EPQEvBTUvFg8TA4M
GBAMCAQEBAQQHHBAKCQcDAwECAQEDAwQFBgYHCAcJCAkICQkJCAkIBwgHBgYFBAMDAgECBQUHCQk
KDAwNDQ4ODw4dHBoiJv4aJgQCBAYGCAkLDA8ICAkJCgoLDAkKCQkJCAkIEA4ODQwLCQkHBgUEEwM
CAgcGBQUDAwIBAgIDBAQEBQUGBgYHBwcHBgcGBgYFBQUEAwMCAgEBAQICBAUFBgcHBQIDAgMDJgc
HBwcGBwYGCwsJBwv+oAMCAQEBBAUHChEVGRwVFhYWFxYgHxwTEBAODQUGBAUDAVwHBgUCAQIDBAU
DpAMEBggKCw0PCAkJCQkKCwsLCwwMDQ0NDQ0MCwsLCgkJCRAODAsJCAYFAwIB3AYFBgYGBgYGDQ0
tGhMVFwwMDQ4OCwsLCwoLCgoJCQgIBwYGBAQDAQEBAgMEBggJCwwOGfEVFRMSEhAQDg4NDAsKCgk
IDwsKCglwChgODg8ODw4NDAoFBAMDAwEBAQEBAgMDBQUFDRARExQWFxgYGBkYBhMdGBQPBQYGBgg
ICAkHBwcGBgYFBQQEBAMCAgEBAgIDBAQEBQUGBgYHBwcICAgHBwcGBRoaFBUXDA0NKAcFBAQCAgE
BAgQEB/6gBAQFBgYNDxASEh4gISEXFhYUExIYFBAIBwQCAgICBAFbCQsNBggHCAcICAgEoxgdHh4
eHh0cGgwMCwsKCQgIBwYFAwMCAQEBAgIDBAQFBgYMDxARERISEhIRAAAAAAUAAAAAA/QD5AA5AI4
AswDaAXcAAAEzHw8VDwcvBj0BLxUlHxMDDwUvFz8BHwk/BTUvDDclHwclLws1PwYfBicXDwQvCzU
/DTMfAycPDh8KDwQdAR8XOwE/CBMfAx0BHw07AT8NPQIvLCMPAQMxBwgODg0LCwoJCAcGBQQDAgE
BAQIDAwQEBAUEAwQCAgICBAMFCAQFBgUGBwcICQgKGxwcHh8V/sMGBg8SExUXFxkgIB8fHx4dHBs
SF+IDBQoJCgsMDg4QEBESExQUFRUWHBkiHRkUDwsHAwEBbiAaGxwdHR4eCAgIBwYGAgIBAgMFBSA
fHh0bGhgfWAEXBi0dIh8aFg/+1yAZHQ0LCQgHBQQDAgEDAwQJDhEREXUXGBoc6QUJCQcGfxMPDg0
LCgkHBgUDAgEDBAMFBQYHCAgJDxAREhAQIyS5Dw4NDAsJCAcFAwMDAQEBAgMGCAoLDRcaExh0BAM
CAQICAwQJDA8REXUXFxoaGhIkJCMhIR8ODg4MCwsQDwkHBgYE1wMDAgEBAgMEBAUFBgcHBwgICAk
ICQgHCAcGBgYFBAMDAgICAgMFBQYGCAgKCgoMDBIJBwgKCwsNDQ4QDyMkJEMdHhwdHBwcGxoaGRg
XFxYWFBQCTAEDBAcICgsMDQ0PDxAQEBFtBAUDAwMCAQEBAQIDAwMFBDcsIAsWFgoKCgcHBgUFBAM
CBQQGCQsIqQoKExQTFBITERUSEA8NCwkGBQIF/ncEAwQBAQEBAQIDBAUGBwgJCgsMERAZGRcWFRI
QCgcGvxoWFBUTFBISAwEBAwUGBgYGBgYFBAQTFBQTFBUUG5gjAxkRGBgYFhIVFhUbDQ0LDAoKCgg
HBgUFAwIEAgEBAwQGBwoLWwEFBwgH3BUSEREQEBAODw0NDAwKCgkGBQUEBAQDAwMCAgEBBAZBBQU
HBwkJCgsICAgJCAkSExMTFBUUFR8gFRnJCAgICQkJCQkJChMUFBQTFBMSEhEQChMQDQwIBgIBAQI
EBgUFBgUGAXQTExYWNwgJCAgHBwYGBgUEAwMCAgICAwMEBQYGBgcHCAgJCG0RERAQEBAPDw4ODQw
LCgkKEwwNDQwNDQ0NDQ0ZGBUiDg4MCwsKCAgHBQUEAwEBAgMADAAAAAAD8gP0AAgADAAQABQAGAA
cAEQASABMAFAAVABYAAATFSE1JwcnBycFMzcjNxc3JwcXNyc/ATUnBxUXNRcVHwg/CD0BLwcrAQ8
HNxc3JwcXNyc7AScjJREhEQMhESF+AwSperIsRwFaCgYWRRwGFp8JHRCZIiLOIhkDBAYICgoGBgc
FDAoKCAYFAgEDBAYICQsGBgYGDAoKCAYFAgFxDxYGrBMPHEgWBgoBEPyuRAPk/BwBr96cVT+yGUs
DIhMWBxwcChYQLgcGBgYGChYJBgsLCQgHBQEBAQECBAcHCgsFBwYGCwsJCAcFAQECBQYICQsGBj8
QHAYGHw8WI1H9BQL7/GMD6AAAAAQAAAAAA/QDqAAGADYAPQBBAAABNxMVITUBJRUfCTsBPwk9AS8
KDwolEQMHAwERAyERIQJJg+v8kgEKAToBAQUHCAoGBQYHBgYGBgYGCQkHBAIBAQIEBwkJBgYGBgY
GBwYFBgoIBwQCAQEq7YL1/vY9A+j8GAFBqf7tQpYBR3oHBgYMCgkHAwICAQECAgMHCQoMBgYHBwY
GDAoJBwMCAQEBAQEBAgMHCQoMBgZ5/cgBF6gBMP64AeH87ANQAAAAAAAAEgDeAAEAAAAAAAAAAQA
AAAEAAAAAAAAEACgABAAEAAAAAAAIABwALAAEAAAAAAAMACgASAAEAAAAAAAQACgAcAAEAAAAAAAU
ACwAmAAEAAAAAAAYACgAxAAEAAAAAAAoALAA7AAEAAAAAAAsAEgBnAAMAAQQJAAAAAgB5AAMAAQQ
JAAEAFAB7AAMAAQQJAAIADgCPAAMAAQQJAAMAFACdAAMAAQQJAAQAFACxAAMAAQQJAAUAFgDFAAM
AAQQJAAYAFADbAAMAAQQJAAoAWADvAAMAAQQJAAsAJAFHIGZvbnQtaWNvbnNSZWd1bGFyZm9udC1
pY29uc2ZvbnQtaWNvbnNWZXJzaW9uIDEuMGZvbnQtaWNvbnNGb250IGdlbmVyYXRlZCB1c2luZyB
TeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAGYAbwBuAHQALQBpAGM
AbwBuAHMAUgBlAGcAdQBsAGEAcgBmAG8AbgB0AC0AaQBjAG8AbgBzAGYAbwBuAHQALQBpAGMAbwB
```

```
uAHMAVgBlAHIAcwBpAG8AbgAgADEALgAwAGYAbwBuAHQALQBpAGMAbwBuAHMARgBvAG4AdAAgAGc
AZQBuAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB
0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMAbwBtAAA
AAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACgECAQMBBAEFAQYBBwEIAQkBCgELAAd
zYXZlXzAyB3NhdmUtMDEHZWRpdF8wMwdlZGl0XzAxBWNsZWFyDHBhaW50LWJ1Y2tldA9wYWludC1
idWNrZXQtd2YGaW1hZ2VzC3BpY3R1cmVzLXdmAAAA) format('truetype');
  font-weight: normal;
  font-style: normal;
}
.e-sign-icons {
  font-family: 'font-icons' !important;
  font-size: 55px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
#signature-toolbar-control {
  border: 1px solid lightgray;
}
.highcontrast #signature-toolbar-control {
  border: 1px solid white;
}
#signature-toolbar-control #toolbar {
  border: none;
  border-bottom: 1px solid lightgray;
  box-sizing: border-box;
}
    #signature-toolbar-control #toolbar {
        height: 44px !important;
    }
#signature-toolbar-control .e-btn:disabled {
  opacity: 0.5 !important;
  pointer-events: none;
}
#signature-toolbar-control #signature-control {
  height: 300px;
  width: 100%;
  margin: 0;
}
#signature-toolbar-control #signature {
  border: none !important;
}
.e-colorpicker-wrapper.e-bg-color #bg-color+.e-split-btn-wrapper .e-split-
btn .e-selected-color {
  background: none;
  border-bottom-style: solid;
  border-bottom-width: 3px;
  width: 14px;
  margin: 0px 2px;
  border-bottom-color: #ffffff;
}
.e-colorpicker-wrapper.e-stroke-color #stroke-color+.e-split-btn-wrapper .e-
split-btn .e-selected-color {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  background: none;
  border-bottom-style: solid;
  border-bottom-width: 3px;
  width: 14px;
  margin: 0px 2px;
  border-bottom-color: #000000;
}
.e-colorpicker-wrapper.e-bg-color #bg-color+.e-split-btn-wrapper .e-split-
btn .e-selected-color .e-split-preview,
.e-colorpicker-wrapper.e-stroke-color #stroke-color+.e-split-btn-wrapper .e-
split-btn .e-selected-color .e-split-preview {
  display: none;
}
.e-colorpicker-wrapper.e-bg-color #bg-color+.e-split-btn-wrapper .e-split-
btn .e-selected-color::before {
  content: '\e707';
}
.e-colorpicker-wrapper.e-stroke-color #stroke-color+.e-split-btn-wrapper .e-
split-btn .e-selected-color::before {
  content: '\e704';
}
#signature-toolbar-control .e-clear::before {
  content: '\e706';
}
#signature-toolbar-control .e-save::before {
  content: '\e701';
}
/* Circle palette customization */
.e-container .e-palette .e-circle-palette {
  border: 0;
  height: 32px;
  width: 32px;
  border-radius: 20px;
  margin: 4px;
}
.e-container .e-palette .e-circle-palette:hover {
  box-shadow: none;
  transform: scale(1.2);
  transition: transform .2s ease-out;
}
.e-circle-palette .e-circle-selection {
  height: 32px;
  width: 32px;
  border-radius: 20px;
  display: inline-block;
  transform: scale(0);
  transition: transform 1.2s ease-in;
}
.e-circle-palette.e-selected .e-circle-selection {
  transform: scale(0.8);
  background-color: #fff;
  transition: transform .2s ease-out;
}
#circle-palette+.e-container,
#scroll-palette+.e-container {
  background-color: transparent;
  border-color: transparent;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  box-shadow: none;
}
.e-container .e-palette .e-circle-palette.e-selected {
    outline: none;
}
.e-bg-color .e-circle-palette.e-nocolor-item.e-selected .e-circle-selection
{
    background: transparent;
}
.e-bg-color .e-circle-palette.e-nocolor-item.e-selected {
    border: 3px solid lightgray;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/signature/default-cs8" %}

Skeleton

Getting Started with the Vue Skeleton Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Skeleton component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies are required to use the Skeleton component in your application.

`js

|-- @syncfusion/ej2-vue-notifications

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-notifications

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

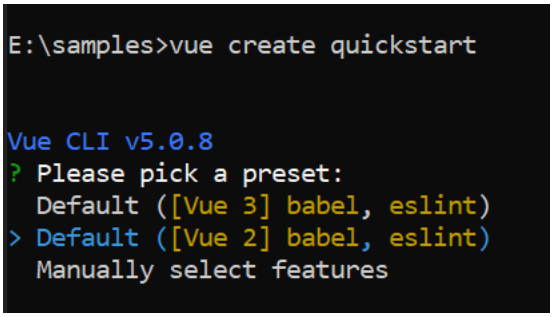
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Skeleton component as an example. Install the @syncfusion/ej2-vue-notifications package by running the following command:

`bash

npm install @syncfusion/ej2-vue-notifications --save

`

or

`bash

yarn add @syncfusion/ej2-vue-notifications

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Skeleton component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-notifications/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Skeleton component using Composition API or Options API:

1\. First, import and register the Skeleton component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SkeletonComponent as EjsSkeleton } from "@syncfusion/ej2-vue-notifications";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SkeletonComponent } from "@syncfusion/ej2-vue-notifications";
export default {
components: {
'ejs-skeleton': SkeletonComponent
}
}
</script>
```

2\. In the template section, define the Skeleton component with the height property.

~/SRC/APP.VUE

```
<template>
<div>
<ejs-skeleton height='15px'></ejs-skeleton>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div>
        <ejs-skeleton height='15px'></ejs-skeleton>
    </div>
</template>
<script setup>
import { SkeletonComponent as EjsSkeleton } from "@syncfusion/ej2-vue-notifications";
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    @import "../node_modules/@syncfusion/ej2-
notifications/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div>
        <ejs-skeleton height='15px'></ejs-skeleton>
    </div>
</template>
<script>
    import { SkeletonComponent } from "@syncfusion/ej2-vue-notifications";
    export default {
        components: {
            'ejs-skeleton': SkeletonComponent
        },
        data: function () {
            return {};
        }
    }
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
notifications/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/skeleton/getting-started-cs1" %}

Getting Started with Syncfusion Skeleton Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Skeleton component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

Copyright © 2001 -2024 Syncfusion Inc.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Skeleton component as an example. To use the Vue Skeleton component in the project, the @syncfusion/ej2-vue-notifications package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-notifications --save

`

or

`bash

yarn add @syncfusion/ej2-vue-notifications

`

Adding CSS reference for Syncfusion Vue Skeleton component

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

Copyright © 2001 -2024 Syncfusion Inc.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Button component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
```

Adding Syncfusion Vue Skeleton component in the application

Follow the below steps to add the Vue Skeleton component using Composition API or Options API:

1. First, import and register the Skeleton component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SkeletonComponent as EjsSkeleton } from "@syncfusion/ej2-vue-
notifications";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SkeletonComponent } from '@syncfusion/ej2-vue-notifications';
// Component registration
export default {
name: "App",
components: {
'ejs-skeleton': SkeletonComponent
}
}
</script>
```

2. Add the component definition in template section.

~/SRC/APP.VUE

```
<template>
<div>
<ejs-skeleton height='15px'></ejs-skeleton>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

Copyright © 2001 -2024 Syncfusion Inc.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div>
<ejs-skeleton height='15px'></ejs-skeleton>
</div>
</template>
<script setup>
import { SkeletonComponent as EjsSkeleton } from "@syncfusion/ej2-vue-
notifications";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div>
<ejs-skeleton height='15px'></ejs-skeleton>
</div>
</template>
<script>
import { SkeletonComponent } from '@syncfusion/ej2-vue-notifications';
// Component registration
export default {
name: "App",
components: {
'ejs-skeleton': SkeletonComponent
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
```

Skeleton Shapes

The shape property can be used to display skeleton in different shapes like circle, square, rectangle and text.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class="control-section">
<div style="width: 200px">
<ejs-skeleton style="margin-right: 30px" shape="Circle" width="48px"></ejs-
skeleton>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<ejs-skeleton shape="Square" width="48px"></ejs-skeleton>
<br /><br />
<ejs-skeleton height="15px"></ejs-skeleton>
<br /><br />
<ejs-skeleton shape="Rectangle" height='50px'></ejs-skeleton>
</div>
</div>
</template>
<script setup>
import { SkeletonComponent as EjsSkeleton} from '@syncfusion/ej2-vue-
notifications';
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
.control-section {
padding: 20px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="control-section">
<div style="width: 200px">
<ejs-skeleton style="margin-right: 30px" shape="Circle" width="48px"></ejs-
skeleton>
<ejs-skeleton shape="Square" width="48px"></ejs-skeleton>
<br /><br />
<ejs-skeleton height="15px"></ejs-skeleton>
<br /><br />
<ejs-skeleton shape="Rectangle" height='50px'></ejs-skeleton>
</div>
</div>
</template>
<script>
import { SkeletonComponent } from '@syncfusion/ej2-vue-notifications';
// Component registration
export default {
name: "App",
components: {
'ejs-skeleton': SkeletonComponent
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
.control-section {
padding: 20px;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Shapes in Vue Skeleton component

The Skeleton control support various built-in shape variants to design layout of the page. You can use the shape property to create a preview of any layout.

The Skeleton component supports the following content shapes:

Circle skeleton shape

`

<template>

<ejs-skeleton shape= 'Circle' width= "48px"></ejs-skeleton>

</template>

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { SkeletonPlugin } from "@syncfusion/ej2-vue-notifications";
Vue.use(SkeletonPlugin);
export default { }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-notifications/styles/material.css";
</style>
`
```

Square skeleton shape

```
`
<template>
<div>
<ejs-skeleton shape= 'Square' width= "48px"></ejs-skeleton>
</div>
</template>
<script>
import Vue from 'vue';
import { SkeletonPlugin } from "@syncfusion/ej2-vue-notifications";
Vue.use(SkeletonPlugin);
export default { }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-notifications/styles/material.css";
</style>
`
```

Rectangle skeleton shape

```
`
<template>
<div>
<ejs-skeleton shape= 'Rectangle' width= "50px"></ejs-skeleton>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</template>
<script>
import Vue from 'vue';
import { SkeletonPlugin } from "@syncfusion/ej2-vue-notifications";
Vue.use(SkeletonPlugin);
export default { }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-notifications/styles/material.css";
</style>
`
```

Text skeleton shape

```
`
<template>
<div>
<ejs-skeleton shape= 'Text' width= "15px"></ejs-skeleton>
</div>
</template>
<script>
import Vue from 'vue';
import { SkeletonPlugin } from "@syncfusion/ej2-vue-notifications";
Vue.use(SkeletonPlugin);
export default { }
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-notifications/styles/material.css";
</style>
`
```

Below example demonstrates the above functionalities of a Skeleton component.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div>
        <div id="skeletonCard">
            <div class='cardProfile'>
                <ejs-skeleton id="cardProfile" shape="Circle"
width="60px"></ejs-skeleton>
            </div>
            <div class="cardinfo">
                <ejs-skeleton id="text1" width="30%" height='15px'></ejs-
skeleton><br/>
                <ejs-skeleton id="text2" width="15%" height='15px'></ejs-
skeleton>
            </div>
            <div class="cardContent">
                <ejs-skeleton id="cardImage" shape="Rectangle" width="100%"
height='150px'></ejs-skeleton>
            </div>
            <div class="cardoptions">
                <ejs-skeleton id="rightOption" shape="Rectangle" width="20%"
height='32px'></ejs-skeleton>
                <ejs-skeleton id="leftOption" shape="Rectangle" width="20%"
height='32px'></ejs-skeleton>
            </div>
        </div>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SkeletonPlugin } from "@syncfusion/ej2-vue-notifications";
    Vue.use(SkeletonPlugin);
    export default {
        data: function () {
            return {};
        }
    }
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
notifications/styles/material.css";
    #skeletonCard {
      padding: 10px;
      line-height: inherit;
      height: 330px;
    }
    #skeletonCard .cardProfile {
      float: left;
      margin-right: 15px;
    }
    #skeletonCard .cardinfo {
      margin-top: 10px;
      overflow: hidden;
    }
    #skeletonCard .cardContent {
      margin: 20px 0px 20px;
    }
    #skeletonCard .cardoptions {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      display: flex;
      justify-content: space-between;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/skeleton/shapes-cs1" %}

Shimmer effect in Vue Skeleton component

You can use the shimmerEffect property to change animation effect in the skeleton control. Skeleton supports Wave, Pulse and Fade effects and by default, the shimmerEffect is set to Wave effect.

`

<template>

<ejs-skeleton shape= 'Circle' width= "60px" shimmerEffect= 'Pulse'></ejs-skeleton>

</template>

<script>

import Vue from 'vue';

import { SkeletonPlugin } from "@syncfusion/ej2-vue-notifications";

Vue.use(SkeletonPlugin);

export default { }

</script>

<style>

@import "../node_modules/@syncfusion/ej2-base/styles/material.css";

@import "../node_modules/@syncfusion/ej2-notifications/styles/material.css";

</style>

`

Below example demonstrates a list with pulse effect skeleton.

APP.VUE

```
<template>
    <div>
        <ul id="skeleton-list" class="e-card">
            <li>
                <div class='listProfile'>
                    <ejs-skeleton shape="Circle" width="40px"
shimmerEffect='Pulse'></ejs-skeleton>
                </div>
                <div>
                    <ejs-skeleton width="60%" height='15px'
shimmerEffect='Pulse'></ejs-skeleton><br>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <ejs-skeleton width="40%" height='15px'
shimmerEffect='Pulse'></ejs-skeleton>
                </div>
            </li>
            <li>
                <div class='listProfile'>
                    <ejs-skeleton shape="Circle" width="40px"
shimmerEffect='Pulse'></ejs-skeleton>
                </div>
                <div>
                    <ejs-skeleton width="60%" height='15px'
shimmerEffect='Pulse'></ejs-skeleton><br>
                    <ejs-skeleton width="40%" height='15px'
shimmerEffect='Pulse'></ejs-skeleton>
                </div>
            </li>
        </ul>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SkeletonPlugin } from "@syncfusion/ej2-vue-notifications";
    Vue.use(SkeletonPlugin);
    export default {
        data: function () {
            return {};
        }
    }
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-
notifications/styles/material.css";
    #skeleton-list {
    padding-left: 12px;
    padding-top: 7px;
    line-height: inherit;
    }
    #skeleton-list li {
        list-style: none;
        display: flow-root;
        margin-bottom: 9px;
    }
    .listProfile {
        float: left;
        margin-right: 15px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/skeleton/effects-cs1" %}

Styles in Vue Skeleton component

You can customize skeleton control in the below ways.

Copyright © 2001 -2024 Syncfusion Inc.

cssClass

You can customize the style of a Skeleton control by using cssClass. The appearance of Vue Skeleton can be customized by changing the wave color, background color, width, and height. For detailed information, refer index.css file below.

APP.VUE

```
<template>
    <div>
        <ejs-skeleton shape= 'Circle' width= "60px" cssClass= "e-customize"></ejs-skeleton>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SkeletonPlugin } from "@syncfusion/ej2-vue-notifications";
    Vue.use(SkeletonPlugin);
    export default { }
</script>
<style>
    @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
    @import "../node_modules/@syncfusion/ej2-notifications/styles/material.css";
    .e-customize.e-skeleton.e-shimmer-wave::after{
    background-image: linear-gradient(90deg, transparent calc(50% - 100px),
    rgb(30 128 234 / 50%) 50%,
    transparent calc(50% + 100px));
    }
    .e-customize.e-skeleton.e-skeleton-circle{
        background-color: #a8c1f2;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/skeleton/styles-cs1" %}

Visible

You can use the visible property which defines the visible state of Skeleton.

`

<template>

<ejs-skeleton shape= 'Circle' width= "60px" visible='false'></ejs-skeleton>

</template>

<script>

import Vue from 'vue';

import { SkeletonPlugin } from "@syncfusion/ej2-vue-notifications";

Vue.use(SkeletonPlugin);

export default { }

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-notifications/styles/material.css";
</style>
`
```

Accessibility in Vue Skeleton component

Accessibility is achieved in the Skeleton component through WAI-ARIA standard. The Skeleton component can be effectively accessed through assistive technologies such as screen readers.

ARIA attributes

The Skeleton component characterized with complete ARIA accessibility support that helps to be accessible by on-screen readers and other assistive technology devices. The following ARIA attributes are applicable for Skeleton component based on its state.

Properties	Functionality
role	This attribute is added to the input element to describe the actual role.
aria-label	Attribute provides the text label with some default description for the Skeleton.
aria-live	The aria-live attribute indicates the priority of updates to a live region.
aria-busy	This attribute is set to true when component is shown.

Smith Chart

Getting Started with the Vue Smithchart Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Smithchart component

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of minimum dependencies are required to use the Smith chart:

```
`javascript
|-- @syncfusion/ej2-vue-charts
|-- @syncfusion/ej2-charts
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-svg-base
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-compression
```

Copyright © 2001 -2024 Syncfusion Inc.

|-- @syncfusion/ej2-file-utils

|-- @syncfusion/ej2-vue-base

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

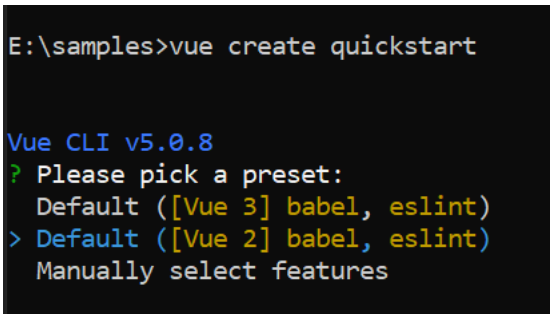
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option `Default ([Vue 2] babel, eslint)` from the menu.



Once the `quickstart` project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Smithchart component as an example. Install the `@syncfusion/ej2-vue-charts` package by running the following command:

`bash

npm install @syncfusion/ej2-vue-charts --save

`

or

`bash

yarn add @syncfusion/ej2-vue-charts

`

Add Syncfusion Vue component

Follow the below steps to add the Vue Smithchart component:

1\. First, import and register the Smithchart component in the script section of the **src/App.vue** file.

~SRC/APP.VUE

```
<script>
import { SmithchartComponent } from "@syncfusion/ej2-vue-charts";
export default {
components: {
'ejs-smithchart': SmithchartComponent
}
}
</script>
```

2\. In the template section, define the Smithchart component.

~SRC/APP.VUE

```
<template>
<div id="app">
<ejs-smithchart></ejs-smithchart>
</div>
</template>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

The following example shows a basic Smith chart.

`

<template>

Copyright © 2001 -2024 Syncfusion Inc.

```
<ejs-smithchart id="smithchart"></ejs-smithchart>
</template>
<script>
import { SmithchartComponent } from '@syncfusion/ej2-vue-charts';
export default {
components: {
'ejs-smithchart': SmithchartComponent
}
}
</script>
`
```

Module Injection

The smith chart component is segregated into individual feature-wise modules. To use a particular feature, inject its feature module using the provide option on component creation. The module available in smith chart and its descriptions is as follows.

- SmithchartLegend - Inject this provider to use legend feature.
- TooltipRender - Inject this provider to use tooltip feature.

In the current application, the above basic Smith chart is modified to visualize transmission lines. For this application, the tooltip and legend features of the Smith chart are used. The feature modules available in the Smith chart and their descriptions are as follows.

Now, import the above mentioned modules from the Smithchart package, and inject it into the Smith chart control using the provide option.

```
`
<template>
<div class="control_wrapper">
<ejs-smithchart id="smithchart"></ejs-smithchart>
</div>
</template>
<script>
import { SmithchartComponent, SmithchartLegend, TooltipRender } from "@syncfusion/ej2-vue-charts";
export default {
components: {
'ejs-smithchart': SmithchartComponent
},
```

Copyright © 2001 -2024 Syncfusion Inc.

data: function() {

return {

}

},

provide:{

smithchart:[Legend, TooltipRender]

}

}

</script>

`

Add series to Smith chart

The Smith chart has the following two type of specification for adding series:

- dataSource - Data object can be bound directly by specifying the resistance and reactance values.
- points - Collection of resistance and reactance values can be bound directly to render the series.

The following two ways demonstrate adding two series to the Smith chart.

- First series Transmission1 shows dataSource bound series.
- Second series Transmission2 shows points bound series.

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart">
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import { SmithchartComponent, SeriesDirective, SeriesCollectionDirective }
from "@syncfusion/ej2-vue-charts";
export default {
  components: {
    'ejs-smithchart': SmithchartComponent,
    'e-series': SeriesDirective,
    'e-seriesCollection': SeriesCollectionDirective
  },
  data: function() {
    return {
      dataSource: [
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
            ],
            name: 'Transmission1',
            reactance: 'reactance', resistance: 'resistance',
            points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0,
reactance: 0.15 },
            { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
            { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
            { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
}],
            name2: 'Transmission2'
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs1" %}

Add title to Smith chart

The following APIs are used in the Smith chart:

- `title` - Adds title to the Smith chart.

Copyright © 2001 -2024 Syncfusion Inc.

- text - Sets text to the title.
- visible - Toggles the title.

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :title='title'>
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import { SmithchartComponent, SeriesDirective, SeriesCollectionDirective }
from "@syncfusion/ej2-vue-charts";
export default {
  components: {
    'ejs-smithchart': SmithchartComponent,
    'e-series': SeriesDirective,
    'e-seriesCollection': SeriesCollectionDirective
  },
  data: function() {
    return {
      title: { text: 'Transmission lines applied for both impedance and
admittance'},
      dataSource: [
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
            ],
            name: 'Transmission1',
            reactance: 'reactance', resistance: 'resistance',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0,
reactance: 0.15 },
            { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
            { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
            { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
}],
            name2: 'Transmission2'
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs2" %}

Enable marker to the Smith chart

To use marker in series and its customization in Smith chart, use series `marker` API. To display marker for a particular series, specify the `marker visible` to true. The following sample marker is enabled for the first series only.

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart">
                <e-seriesCollection>
                    <e-series :title='title' :dataSource='dataSource'
:name='name' :reactance='reactance' :resistance='resistance'
:marker='marker'></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import { SmithchartComponent, SeriesDirective, SeriesCollectionDirective }
from "@syncfusion/ej2-vue-charts";
export default {
  components: {
    'ejs-smithchart': SmithchartComponent,
    'e-series': SeriesDirective,
    'e-seriesCollection': SeriesCollectionDirective
  },
  data: function() {
    return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        title: { text: 'Transmission lines applied for both impedance and
admittance'},
        dataSource: [
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
            ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        marker: {
                visible: true
        },
        points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0,
reactance: 0.15 },
            { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
            { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
            { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
}],
            name2: 'Transmission2'
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs3" %}

Copyright © 2001 -2024 Syncfusion Inc.

Enable data label to marker

To use marker data label and its customization in Smith chart, use marker dataLabel. To display data label for a particular series marker, specify the dataLabel visible to true in that series marker. The following sample data label is enabled for the first series.

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart">
                <e-seriesCollection>
                    <e-series :title='title' :dataSource='dataSource'
:name='name' :reactance='reactance' :resistance='resistance'
:marker='marker'></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import { SmithchartComponent, SeriesDirective, SeriesCollectionDirective }
from "@syncfusion/ej2-vue-charts";
export default {
  components: {
    'ejs-smithchart': SmithchartComponent,
    'e-series': SeriesDirective,
    'e-seriesCollection': SeriesCollectionDirective
  },
  data: function() {
    return {
        title: { text: 'Transmission lines applied for both impedance and
admittance'},
        dataSource: [
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
            ],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        marker: {
                visible: true,
                dataLabel: {
                    visible: true
                }
            },
        points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0,
reactance: 0.15 },
            { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
            { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
            { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
}],
        name2: 'Transmission2'
        }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs4" %}

Enable legend for Smith chart

The legend feature is used to denote the corresponding series. To enable the legend, inject the SmithchartLegend module using the provide option and set visible to true in legendSettings. The following sample shows enabling legend for Smith chart. The series name can be customized using the series name.

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :legendSettings='legendSettings'>
                <e-seriesCollection>
                    <e-series :title='title' :dataSource='dataSource'
:name='name' :reactance='reactance' :resistance='resistance'
marker='marker'></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { SmithchartComponent, SeriesDirective, SeriesCollectionDirective,
SmithchartLegend } from "@syncfusion/ej2-vue-charts";
export default {
  components: {
    'ejs-smithchart': SmithchartComponent,
    'e-series': SeriesDirective,
    'e-seriesCollection': SeriesCollectionDirective
  },
  data: function() {
    return {
        title: { text: 'Transmission lines applied for both impedance and
admittance'},
        legendSettings: {
        visible: true
            },
        dataSource: [
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
             ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        marker: {
                visible: true,
                dataLabel: {
                    visible: true
                }
        },
        points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0,
reactance: 0.15 },
            { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
```

```
            { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
            { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
}],
        name2: 'Transmission2'
    }
  },
provide:{
    smithchart:[SmithchartLegend]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs5" %}

Enable tooltip for the Smith chart series

The tooltip feature is used to show the values of the current point. To enable the tooltip, inject the TooltipRender module using the provide option and set visible to true. The following sample shows enabling tooltip for Smith chart series collection.

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :legendSettings='legendSettings'>
                <e-seriesCollection>
                    <e-series :title='title' :dataSource='dataSource'
:tooltip='tooltip' :name='name' :reactance='reactance'
:resistance='resistance' marker='marker'></e-series>
                    <e-series :points='points' :name='name2'
:tooltip='tooltip2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import { SmithchartComponent, SeriesDirective, SeriesCollectionDirective,
SmithchartLegend,TooltipRender } from "@syncfusion/ej2-vue-charts";
export default {
  components: {
    'ejs-smithchart': SmithchartComponent,
    'e-series': SeriesDirective,
    'e-seriesCollection': SeriesCollectionDirective
  },
  data: function() {
    return {
            title: { text: 'Transmission lines applied for both impedance
and admittance'},
            legendSettings: {
                visible: true
            },
            dataSource: [
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
            ],
            name: 'Transmission1',
            reactance: 'reactance', resistance: 'resistance',
            tooltip: {
                visible: true
            },
            marker: {
                visible: true,
                dataLabel: {
                    visible: true
                }
            },
            points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0,
reactance: 0.15 },
            { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
            { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
            { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
            { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
}],
            name2: 'Transmission2',
            tooltip2: {
                visible: true
            }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
   },
provide:{
    smithchart:[SmithchartLegend,TooltipRender]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs6" %}

Getting Started with the Vue Smith Chart Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Smith Chart component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Copyright © 2001 -2024 Syncfusion Inc.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Smith Chart component as an example. To use the Vue Smith Chart component in the project, the @syncfusion/ej2-vue-charts package needs to be installed using the following command:

Copyright © 2001 -2024 Syncfusion Inc.

`bash

npm install @syncfusion/ej2-vue-charts --save

`

or

`bash

yarn add @syncfusion/ej2-vue-charts

`

Add Syncfusion Vue component

Follow the below steps to add the Vue Smith Chart component using Composition API or Options API:

1.First, import and register the Smith Chart component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SmithchartComponent as EjsSmithchart, SmithchartLegend,
TooltipRender, SeriesCollectionDirective as ESeriesCollection,
SeriesDirective as ESeries } from "@syncfusion/ej2-vue-charts";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SmithchartComponent, SmithchartLegend, TooltipRender,
SeriesCollectionDirective, SeriesDirective } from "@syncfusion/ej2-vue-
charts";
//Component registration
export default {
name: "App",
components: {
"ejs-smithchart": SmithchartComponent,
"e-seriesCollection": SeriesCollectionDirective,
"e-series" : SeriesDirective
}
}
</script>
```

2.In the template section, define the Smith Chart component with the [dataSource] property.

~/SRC/APP.VUE

```
<template>
<ejs-smithchart id="smithchart" :title='title'
:legendSettings='legendSettings'>
<e-seriesCollection>
<e-series :points='dataSource' :marker='marker' :tooltip='tooltip'
:name='name' :reactance='reactance' :resistance='resistance'></e-series>
<e-series :points='points' :tooltip='tooltip' :name='name1'></e-series>
</e-seriesCollection>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</ejs-smithchart>
</template>
```

3.Declare the values for the dataSource property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const dataSource = [
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0.3, reactance: 0.1 }, { resistance: 0.5, reactance: 0.2 },
{ resistance: 1.5, reactance: 0.5 }, { resistance: 2.0, reactance: 0.5 },
{ resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
{ resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
{ resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
{ resistance: 4.5, reactance: -0.5 }, { resistance: 5.0, reactance: -1.0 }
];
const points = [
{ resistance: 0, reactance: 0.15 }, { resistance: 0, reactance: 0.15 },
{ resistance: 0, reactance: 0.15 }, { resistance: 0.3, reactance: 0.2 },
{ resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance: 0.2 },
{ resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance: 0.2 },
{ resistance: 0.5, reactance: 0.4 }, { resistance: 1.0, reactance: 0.8 },
{ resistance: 2.5, reactance: 1.3 }, { resistance: 3.5, reactance: 1.6 },
{ resistance: 3.5, reactance: 1.6 }, { resistance: 3.5, reactance: 1.6 },
{ resistance: 4.5, reactance: 2.0 }, { resistance: 6.0, reactance: 4.5 },
{ resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25 }
];
const tooltip = { visible: true };
const legendSettings = { visible: true };
const marker = {
visible: true,
dataLabel: {
visible: true
}
};
const title = { text: 'Transmission lines applied for both impedance and
admittance'};
const name = 'Transmission1';
const name1 = 'Transmission2';
const reactance = 'reactance';
const resistance = 'resistance';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
dataSource: [
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0.3, reactance: 0.1 }, { resistance: 0.5, reactance: 0.2 },
{ resistance: 1.5, reactance: 0.5 }, { resistance: 2.0, reactance: 0.5 },
{ resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
{ resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
{ resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
{ resistance: 4.5, reactance: -0.5 }, { resistance: 5.0, reactance: -1.0 }
],
points: [
{ resistance: 0, reactance: 0.15 }, { resistance: 0, reactance: 0.15 },
{ resistance: 0, reactance: 0.15 }, { resistance: 0.3, reactance: 0.2 },
{ resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance: 0.2 },
{ resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance: 0.2 },
{ resistance: 0.5, reactance: 0.4 }, { resistance: 1.0, reactance: 0.8 },
{ resistance: 2.5, reactance: 1.3 }, { resistance: 3.5, reactance: 1.6 },
{ resistance: 3.5, reactance: 1.6 }, { resistance: 3.5, reactance: 1.6 },
{ resistance: 4.5, reactance: 2.0 }, { resistance: 6.0, reactance: 4.5 },
{ resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25 }
],
tooltip: {
visible: true
},
legendSettings: {
visible: true
},
marker: {
visible: true,
dataLabel: {
visible: true
}
},
title: { text: 'Transmission lines applied for both impedance and
admittance'},
name: 'Transmission1',
name1: 'Transmission2',
reactance: 'reactance',
resistance: 'resistance'
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-smithchart id="smithchart" :title='title'
:legendSettings='legendSettings'>
<e-seriesCollection>
<e-series :points='dataSource' :marker='marker' :tooltip='tooltip'
:name='name' :reactance='reactance' :resistance='resistance'></e-series>
<e-series :points='points' :tooltip='tooltip' :name='name1'></e-series>
</e-seriesCollection>
</ejs-smithchart>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script setup>
import { provide } from 'vue';
import { SmithchartComponent as EjsSmithchart, SmithchartLegend,
TooltipRender, SeriesCollectionDirective as ESeriesCollection,
SeriesDirective as ESeries } from "@syncfusion/ej2-vue-charts";
const dataSource = [
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0.3, reactance: 0.1 }, { resistance: 0.5, reactance: 0.2 },
{ resistance: 1.5, reactance: 0.5 }, { resistance: 2.0, reactance: 0.5 },
{ resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
{ resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
{ resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
{ resistance: 4.5, reactance: -0.5 }, { resistance: 5.0, reactance: -1.0 }
];
const points = [
{ resistance: 0, reactance: 0.15 }, { resistance: 0, reactance: 0.15 },
{ resistance: 0, reactance: 0.15 }, { resistance: 0.3, reactance: 0.2 },
{ resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance: 0.2 },
{ resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance: 0.2 },
{ resistance: 0.5, reactance: 0.4 }, { resistance: 1.0, reactance: 0.8 },
{ resistance: 2.5, reactance: 1.3 }, { resistance: 3.5, reactance: 1.6 },
{ resistance: 3.5, reactance: 1.6 }, { resistance: 3.5, reactance: 1.6 },
{ resistance: 4.5, reactance: 2.0 }, { resistance: 6.0, reactance: 4.5 },
{ resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25 }
];
const tooltip = { visible: true };
const legendSettings = { visible: true };
const marker = {
visible: true,
dataLabel: {
visible: true
}
};
const title = { text: 'Transmission lines applied for both impedance and
admittance'};
const name = 'Transmission1';
const name1 = 'Transmission2';
const reactance = 'reactance';
const resistance = 'resistance';
const smithchart = [SmithchartLegend, TooltipRender];
provide('smithchart', smithchart);
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-smithchart id="smithchart" :title='title'
:legendSettings='legendSettings'>
<e-seriesCollection>
<e-series :points='dataSource' :marker='marker' :tooltip='tooltip'
:name='name' :reactance='reactance' :resistance='resistance'></e-series>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-series :points='points' :tooltip='tooltip' :name='name1'></e-series>
</e-seriesCollection>
</ejs-smithchart>
</template>
<script>
import { SmithchartComponent, SmithchartLegend, TooltipRender,
SeriesCollectionDirective, SeriesDirective } from "@syncfusion/ej2-vue-
charts";
//Component registration
export default {
name: "App",
components: {
"ejs-smithchart": SmithchartComponent,
"e-seriesCollection": SeriesCollectionDirective,
"e-series" : SeriesDirective
},
data() {
return {
dataSource: [
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
{ resistance: 0.3, reactance: 0.1 }, { resistance: 0.5, reactance: 0.2 },
{ resistance: 1.5, reactance: 0.5 }, { resistance: 2.0, reactance: 0.5 },
{ resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
{ resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
{ resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
{ resistance: 4.5, reactance: -0.5 }, { resistance: 5.0, reactance: -1.0 }
],
points: [
{ resistance: 0, reactance: 0.15 }, { resistance: 0, reactance: 0.15 },
{ resistance: 0, reactance: 0.15 }, { resistance: 0.3, reactance: 0.2 },
{ resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance: 0.2 },
{ resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance: 0.2 },
{ resistance: 0.5, reactance: 0.4 }, { resistance: 1.0, reactance: 0.8 },
{ resistance: 2.5, reactance: 1.3 }, { resistance: 3.5, reactance: 1.6 },
{ resistance: 3.5, reactance: 1.6 }, { resistance: 3.5, reactance: 1.6 },
{ resistance: 4.5, reactance: 2.0 }, { resistance: 6.0, reactance: 4.5 },
{ resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25 }
],
tooltip: {
visible: true
},
legendSettings: {
visible: true
},
marker: {
visible: true,
dataLabel: {
visible: true
}
},
title: { text: 'Transmission lines applied for both impedance and
admittance'},
name: 'Transmission1',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
name1: 'Transmission2',
reactance: 'reactance',
resistance: 'resistance'
};
},
provide: {
smithchart:[SmithchartLegend, TooltipRender]
},
};
</script>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

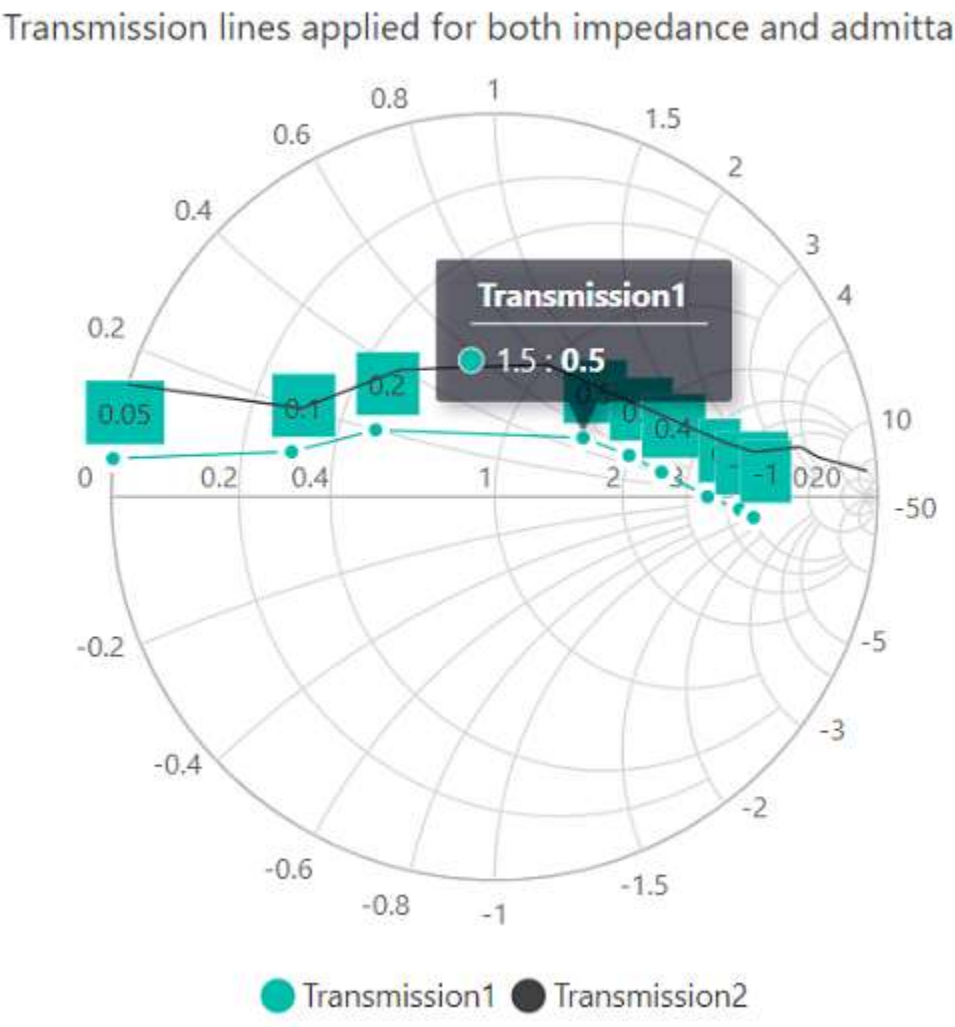
or

`bash

yarn run dev

`

The output will appear as follows:



Sample: vue-3-smith-chart-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Working with data in Vue Smithchart component

Smithchart can visualise the data bound from local data. The data you bind for the smithchart, should be an array of object and that should contain the field resistance and rectangle. This should be bind to points or datasource in the smithchart.

Data Binding

You can bind simple JSON data to smithchart using point property in series. JSON data should contain [resistance] and [reactance] fields. This JSON data should be bind to points or datasource in the smithchart. You can any number of JSON for points or datasource as per your requirement.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" >
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance' ></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        dataSource: [
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
                { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
             ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0,
reactance: 0.15 },
             { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
             { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
             { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
             { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
            { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
            { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
}],
        name2: 'Transmission2'
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs29" %}

Smithchart dimensions in Vue Smithchart component

You can render the smithchart either corresponding to its container size or you can set the size of the smithchart as per your requirement. To render the smithchart corresponding to its container size, you need to set the size for the smithchart container. Else to set the size for the smithchart as per your requirement, you can use the width and height properties in the smithchart.

Size for Container

You can render smithchart to it's container size. To achieve this, you need to specify the width and height of the smithchart's container via inline or CSS as demonstrated below.

`javascript

`

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart">
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
      dataSource: [
```

```
                    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                  { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                  { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                  { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                  { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                  { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 }
              ],
              name: 'Transmission1',
              reactance: 'reactance', resistance: 'resistance',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs10" %}

Size for Smithchart

<!-- markdownlint-disable MD036 -->

You can also set size for smithchart directly through [width] and [height] properties. Using this properties, you can directly mention the width and height of the smithchart in pixels or you can set the width and height in percentage.

In Pixel

In smithchart's width and height property, you can directly give values in pixels like below demonstration. This will render smithchart in same size as you mentioned in you code.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :width='width' :height='height'>
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        width: '650px',
        height: '300px',
        dataSource: [
```

```
                    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                  { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                  { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                  { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                  { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                  { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
              ],
              name: 'Transmission1',
              reactance: 'reactance', resistance: 'resistance',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs11" %}

In percentage

You can also specify the width and height of the smithchart in percentage. If you mention the width and height in percentage, then smithchart will be render as per the percentage of it's container size. You can set the values in percentage like below demonstration.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :width='width' :height='height'>
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        width: '90%',
        height: '80%',
        dataSource: [
                    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                  { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                  { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
```

```
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ],
            name: 'Transmission1',
            reactance: 'reactance', resistance: 'resistance',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs12" %}

Title subtitle in Vue Smithchart component

Enable title

Title and subtitle is used to depicts the information about the data plotted in the smithchart. You can set the title and subtitle of the smithchart using the [text] property in title and subtitle. By default visibility of the title as well as subtitle is enabled. You need to set simply text for title and subtitle in your sample as like below.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :title='title'>
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        title : {
        text: 'Impedance Transmission',
        subtitle: {
            text: 'Transmission'
        }
    },
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ],
            name: 'Transmission1',
            reactance: 'reactance', resistance: 'resistance',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs27" %}

Title trim

Both title and subtitle of the smithchart can be trimmed if it exceeds the certain length. Trimming is enabled using [enableTrim] for title as well as subtitle. This length can be changed using the property [maximumWidth]. Also [font], [textAlignment] and [visibility] can be customized for title as well as subtitle.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart"  :title='title'>
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        title : {
        enableTrim: true,
        maximumWidth: 70,
        textAlignment: 'Center',
        visible: true,
        text: 'Impedance Transmission',
        subtitle: {
            text: 'Transmission',
            visible: true,
            textAlignment: 'Far',
            enableTrim: true,
            maximumWidth: 40
        }
    },
        dataSource: [
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                  { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                  { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                  { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                  { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                  { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
              ],
              name: 'Transmission1',
              reactance: 'reactance', resistance: 'resistance',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs28" %}

Smithchart axis in Vue Smithchart component

Like chart, Smithchart is having support for two types of axis.

- Horizontal axis - axis drawn as straight line in the horizontal direction of the chart.
- Radial axis - axis is drawn as circular path.

Labels Customization

Axis labels are used to denote what kind of data is bound for smithchart. Using axis labels, you can easily identify in which interval chart is rendered. Using following properties we can customize the axis labels for horizontal and radial axis.

- [labelPosition] - used to place the labels either inside or outside the axis line.
- [labelIntersectAction] - used to hide the labels when intersect with other one.
- [labelStyle] - used to customize the properties such as font size, family, weight, opacity.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :horizontalAxis='horizontalAxis'
:radialAxis='radialAxis'>
                <e-seriesCollection>
                    <e-series :points='dataSource' :name='name'
:reactance='reactance' :resistance='resistance' ></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { SmithchartPlugin,SmithchartLegend } from "@syncfusion/ej2-vue-
charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        radialAxis : {
        labelPosition: 'Inside',
        labelIntersectAction: 'Hide',
        labelStyle: {
            fontFamily: 'Times New Roman',
            fontWeight: 'bold',
            fontStyle: 'Italic',
            opacity: 0.75,
            size: '14px'
        }
    },
    horizontalAxis: {
        labelPosition: 'Inside',
        labelIntersectAction: 'Hide',
        labelStyle: {
            fontFamily: 'Times New Roman',
            fontWeight: 'bold',
            fontStyle: 'Italic',
            opacity: 0.75,
            size: '14px'
        }
    },
        dataSource: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        points: [{ resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
                { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },],
        name2: 'Transmission2'
    }
  },
provide:{
    smithchart:[SmithchartLegend]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs7" %}

Gridlines

To make the data in a chart that displays axes easier to read, you can display horizontal and radial axis gridlines. Gridlines extend from any horizontal and radial axes across the plot area of the smithchart.

Both horizontal and radial axis are having support for major as well as minor gridlines. Major gridlines are drawn from the position in which labels are rendered. Minor gridlines are drawn between two major gridlines as per the count we set in settings.

We can customize following things, in major as well as minor gridlines.

- [width] - used to customize the width of gridlines.
- [dashArray] - used to customize whether gridline have to render as normal line or dashed line.
- [visible] - used to enable or disable the visibility of the gridlines.
- [opacity] - used to customize the opacity of the major gridlines.
- [count] - used to customize the count of the minor gridlines.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart"  :horizontalAxis= 'horizontalAxis'>
                <e-seriesCollection>
                    <e-series :points='dataSource' :name='name'
:reactance='reactance' :resistance='resistance' ></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin,SmithchartLegend } from "@syncfusion/ej2-vue-
charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        horizontalAxis: {
        majorGridLines: {
```

```
            visible: true,
            opacity: 0.8,
            width: 10
        }
    },
        dataSource: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        points: [{ resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
                { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
                { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },],
        name2: 'Transmission2'
    }
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs8" %}

Axisline

As name suggests that, it is a line in smithchart that can be configured to denotes the axis. By default, visibility of the axis line is true. You can customize its visibility by using visible property in axis Line. Other than visibility of the axis line, you can customize the following properties of the axis line.

- [width] - used to customize the width of the axis line.
- [dashArray] - used to render the axis line as dashed line.
- [visible] - used to enable or disable the visibility of the axis line.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart"  :horizontalAxis= 'horizontalAxis'>
                <e-seriesCollection>
                    <e-series :points='dataSource' :name='name'
:reactance='reactance' :resistance='resistance' ></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin,SmithchartLegend } from "@syncfusion/ej2-vue-
charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
         horizontalAxis: {
        majorGridLines: {
            visible: true,
            opacity: 0.8,
            width: 10
        },
        axisLine: {
            width: 10,
            visible: false
        }
    },
        dataSource: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        points: [{ resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
                { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
                { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },],
        name2: 'Transmission2'
    }
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs9" %}

<!-- markdownlint-disable MD036 -->

Smithchart legend in Vue Smithchart component

Legend is a key used in smithchart, that contains symbol and descriptions. It provides valuable information for interpreting what the smithchart is displaying and can be represented in various colors, shapes or other identifiers based on the data. In simple words, we can define that legend is used to denote the series rendered in the smithchart.

Position and Alignment

By default visibility of the legend is false. To enable the legend, kindly set visible as true in legendSettings. Default position for the legend is bottom. By using [position] property, you can change the position of the legend. You can either place the legend at bottom, top, right and left side of the smithchart. To use the legend in smithchart, you need to import and inject the SmithchartLegend from chart.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :legendSettings='legendSettings'>
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin,SmithchartLegend } from "@syncfusion/ej2-vue-
charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        legendSettings: {
            visible: true,
            position: 'Top'
```

```
            },
        dataSource: [
                    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        points: [{ resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
                { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
                { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 }],
        name2: 'Transmission2'
    }
  },
provide:{
    smithchart:[SmithchartLegend]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs13" %}

Custom Position

Other than these positions, you can place the legend anywhere in the smithchart. To achieve this, you have to set position as custom in legendSettings and specify the x and y coordinates using the x and y properties in the location.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :legendSettings='legendSettings'>
                <e-seriesCollection>
```

```
                    <e-series :dataSource='dataSource'  :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                    <e-series :points='points'  :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin,SmithchartLegend } from "@syncfusion/ej2-vue-
charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        legendSettings: {
        visible: true,
        position: 'Custom',
        location:{
            x: 80,
            y: 100
        }
    },
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        points: [{ resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
                { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
                { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 }],
        name2: 'Transmission2'
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
provide:{
    smithchart:[SmithchartLegend]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs14" %}

Legend Alignment

Other than positioning the legend in the smithchart, you can customize its alignment also. By default, legend is aligned at center. Using the [alignment] property, you can align the legend in near and far locations of the smithchart.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :legendSettings='legendSettings'>
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin,SmithchartLegend } from "@syncfusion/ej2-vue-
charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        legendSettings: {
        visible: true,
        position: 'Top',
        alignment: 'Near'
    },
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
```

```
        points: [{ resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
                { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
                { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 }],
        name2: 'Transmission2'
    }
  },
provide:{
    smithchart:[SmithchartLegend]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs15" %}

Customization

Legend Shape

By default, legend is rendered in the circle shape and the color of the shape is as same as series color in the smithchart. Using the property [shape] in legend settings, you can change the icon shape of the legend as rectangle, triangle and so on.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :legendSettings='legendSettings'>
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin,SmithchartLegend } from "@syncfusion/ej2-vue-
charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        legendSettings: {
        visible: true,
```

```
        position: 'Top',
        shape: 'Rectangle'
    },
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        points: [{ resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
                { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
                { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 }],
        name2: 'Transmission2'
    }
  },
provide:{
    smithchart:[SmithchartLegend]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs16" %}

Legend Size

By default, legend takes 20% - 25% of the chart's height horizontally, when it is placed on top or bottom position and 20% - 25% of the width vertically, while placing on left or right position of the chart. You can change this default legend size by using the [width] and [height] property of the legendSettings.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :legendSettings='legendSettings'>
```

```
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin,SmithchartLegend } from "@syncfusion/ej2-vue-
charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        legendSettings: {
        visible: true,
        position: 'Top',
        height: 100,
        width: 200
    },
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        points: [{ resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
                { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
                { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 }],
        name2: 'Transmission2'
    }
  },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
provide:{
    smithchart:[SmithchartLegend]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs17" %}

Padding

You can customize the space between two legend items and space between legend shape and text as per your requirement. For customizing the space between two legend items, you can use [itemPadding] property. To control space between legend shape and text, you can use [shapePadding] property.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :legendSettings='legendSettings'>
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin,SmithchartLegend } from "@syncfusion/ej2-vue-
charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        legendSettings: {
       visible: true,
        position: 'Top',
        itemPadding: 5,
        shapePadding: 10
    },
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
        name: 'Transmission1',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        reactance: 'reactance', resistance: 'resistance',
        points: [{ resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
                { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
                { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 }],
        name2: 'Transmission2'
    }
  },
provide:{
    smithchart:[SmithchartLegend]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs18" %}

Toggle Visibility

By default series name is displayed in the legend. You can collapse the visibility of the series by clicking the legend for the particular series. You can toggle the series visibility as true or false using the [toggleVisibility] property. By default it is true.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart" :legendSettings='legendSettings'>
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                    <e-series :points='points' :name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin,SmithchartLegend } from "@syncfusion/ej2-vue-
charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        legendSettings: {
        visible: true,
```

```
        position: 'Top',
        toggleVisibility: false
    },
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        points: [{ resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
                { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
                { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 }],
        name2: 'Transmission2'
    }
  },
provide:{
    smithchart:[SmithchartLegend]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs19" %}

Smithchart tooltip in Vue Smithchart component

Smithchart will display details about the points through tooltip, when the mouse is moved over the point. By default, tooltip is disabled. To enable the tooltip for smithchart, you need to import and inject TooltipRender module from chart. And also set the property visible as true, in tooltip settings. You can customize the tooltip's visibility and appearance differently each series in the smithchart.

APP.VUE

```
<template>
    <div class="control_wrapper">
```

```
        <ejs-smithchart id="smithchart" :legendSettings='legendSettings'>
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:tooltip='tooltip' :reactance='reactance' :resistance='resistance'></e-
series>
                    <e-series :points='points'  :tooltip='tooltip'
:name='name2'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin,TooltipRender } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        legendSettings: {
        visible: true,
        position: 'Custom',
        location:{
            x: 80,
            y: 100
        }
    },
    tooltip: {
        visible: true
    },
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
        name: 'Transmission1',
        reactance: 'reactance', resistance: 'resistance',
        points: [{ resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
                { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 }],
        name2: 'Transmission2'
    }
  },
provide:{
    smithchart:[TooltipRender]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs26" %}

<!-- markdownlint-disable MD036 -->

Smithchart marker in Vue Smithchart component

Markers and Datalabels are used to provide information about the data points in the series. You can add a shape to adorn each data point. By default marker and datalabel both are disabled in smithchart. You can enable both of them by setting visible property as true in marker and datalabel settings

Marker

Default visibility of marker is false. You can enable the marker by setting property visible as true in marker settings. This will add marker for each point in the series. Using marker setting, you can customize marker differently for each series in smithchart.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart"  >
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:marker='marker' :reactance='reactance' :resistance='resistance'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        marker: {
                dataLabel: {
                    visible: true,
                    fill: '#99ffcc',
                    opacity: 1,
                    border: {
                        color: '#1aff8c',
                        width: 2,
                    }
                }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            }
        dataSource: [
                    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
             name: 'Transmission1',
             reactance: 'reactance', resistance: 'resistance',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs20" %}

Marker Customization

Using marker settings in series, you can customize the marker for each series differently. Using marker settings, you can customize following properties differently for each series in the smithchart.

- [width] - To control the width of the marker.
- [height] - To control the height of the marker.
- [fill] - Used to customize the fill color of the marker.
- [opacity] - Used to customize the opacity of the marker.
- [border] - Used to control the width and color of the marker's border.
- [shape] - Used to change the shape of the marker.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart"  >
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:marker='marker' :reactance='reactance' :resistance='resistance'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
```

```
    return {
        marker: {
                visible: true,
                height: 10,
                width: 10,
                fill: '#ff99ff',
                opacity: 1,
                shape: 'rectangle',
                border: {
                    color: '#cc00cc',
                    width: 2
                }
            }
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
             name: 'Transmission1',
             reactance: 'reactance', resistance: 'resistance',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs21" %}

Datalabels

By default, datalabel is disabled. You can enable the datalabel by setting property visible as true in datalabel settings. For each point in series, data label is created. Datalabel for each series can be customized differently using datalabel settings.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart"  >
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:marker='marker' :reactance='reactance' :resistance='resistance'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        marker: {
                dataLabel: {
                    visible: true
                }
            }
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
             name: 'Transmission1',
             reactance: 'reactance', resistance: 'resistance',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs22" %}

Datalabel customization

Using datalabel settings in marker, you can customize the datalabel for each series differently. In datalabel, you can customize the following properties differently for each series.

- [fill] - Used to changes the fill color of the data label's shape.
- [opacity] - Used to control the opacity of the data label's shape.
- [border] - Used to customize the width and color of the border.
- [textStyle] - Used to customize the font color, width and size.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart"  >
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:marker='marker' :reactance='reactance' :resistance='resistance'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        marker: {
                dataLabel: {
                    visible: true,
                    fill: '#99ffcc',
                    opacity: 1,
                    border: {
                        color: '#1aff8c',
                        width: 2,
                    }
                }
            }
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
             name: 'Transmission1',
             reactance: 'reactance', resistance: 'resistance',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs23" %}

Smithchart series in Vue Smithchart component

You can add any number of series to the smithchart as per your requirement. You can use series setting to either add or customize the data. For the points or datasource added in the series, line is drawn. You can customize the each series as per your requirement with marker, datalabel, animation, opacity and so on.

points or datasource

For adding values in the smithchart, you can use either points or datasource in the series. Points and datasource both should be array of object which should contain the field names resistance and rectangle.

APP.VUE

```
<template>
    <div class="control_wrapper">
```

```
        <ejs-smithchart id="smithchart"  >
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:reactance='reactance' :resistance='resistance'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
             ],
             name: 'Transmission1',
             reactance: 'reactance', resistance: 'resistance',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs24" %}

Series customization

Using following options in series settings, you can customize each series in smithchart as per your requirement.

- [fill] - Used to customize the fill color for the series.
- [enableSmartLabels] - Used to place the data labels on the smithchart without overlapping with each other.
- [visibility] - Used to handle the visibility of the series.
- [opacity] - Used to control the opacity of the series line.
- [width] - Used to customize the width of the series line.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-smithchart id="smithchart"  >
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-seriesCollection>
                    <e-series :dataSource='dataSource' :name='name'
:fill='fill' :visibility='visibility':opacity='opacity' :width='width'
:marker='marker' :reactance='reactance' :resistance='resistance'></e-series>
                </e-seriesCollection>
        </ejs-smithchart>
    </div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
  data: function() {
    return {
            fill: '#009933',
            visibility: 'visible',
            opacity: 0.75,
            width: 2.5,
            marker: {
                dataLabel: {
                    visible: true
                }
            }
        dataSource: [
                 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ],
            name: 'Transmission1',
            reactance: 'reactance', resistance: 'resistance',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/smithchart/getting-started-cs25" %}

Smithchart print in Vue Smithchart component

Print

The rendered smithchart can be printed directly from the browser by calling the public method print. ID of the smithchart's div element must be passed as argument to that method.

`

<template>

```
<div class="control_wrapper">
<ejs-smithchart id="smithchart" ref='smithchart' >
<e-seriesCollection>
<e-series :dataSource='dataSource' :name='name' :reactance='reactance' :resistance='resistance'
:loaded='loaded'></e-series>
</e-seriesCollection>
</ejs-smithchart>
</div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
data: function() {
return {
dataSource: [
{ resistance: 10, reactance: 25 }, { resistance: 8, reactance: 6 },
{ resistance: 6, reactance: 4.5 }, { resistance: 4.5, reactance: 2 },
{ resistance: 3.5, reactance: 1.6 }, { resistance: 2.5, reactance: 1.3 },
{ resistance: 2, reactance: 1.2 }, { resistance: 1.5, reactance: 1 },
{ resistance: 1, reactance: 0.8 }, { resistance: 0.5, reactance: 0.4 },
{ resistance: 0.3, reactance: 0.2 }, { resistance: 0, reactance: 0.15 },
],
name: 'Transmission1',
reactance: 'reactance', resistance: 'resistance',
}
},
methods: {
loaded: function(args) {
args.smithchart.print();
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
```

Export

The rendered smithchart can be exported to JPEG , PNG, SVG or PDF format by using export method in smithchart. Input parameters for this method are Export Type for format and fileName of result.

```
<template>
<div class="control_wrapper">
<ejs-smithchart id="smithchart" ref='smithchart'  >
<e-seriesCollection>
<e-series :dataSource='dataSource' :name='name' :reactance='reactance' :resistance='resistance' :loaded='loaded'></e-series>
</e-seriesCollection>
</ejs-smithchart>
</div>
</template>
<script>
import Vue from 'vue';
import { SmithchartPlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SmithchartPlugin);
export default {
data: function() {
return {
dataSource: [
{ resistance: 10, reactance: 25 }, { resistance: 8, reactance: 6 },
{ resistance: 6, reactance: 4.5 }, { resistance: 4.5, reactance: 2 },
{ resistance: 3.5, reactance: 1.6 }, { resistance: 2.5, reactance: 1.3 },
{ resistance: 2, reactance: 1.2 }, { resistance: 1.5, reactance: 1 },
{ resistance: 1, reactance: 0.8 }, { resistance: 0.5, reactance: 0.4 },
{ resistance: 0.3, reactance: 0.2 }, { resistance: 0, reactance: 0.15 },
],
name: 'Transmission1',
reactance: 'reactance', resistance: 'resistance',
```

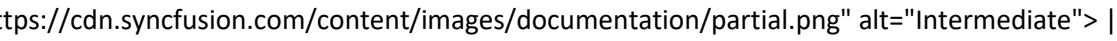
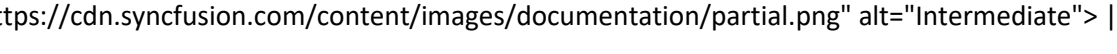
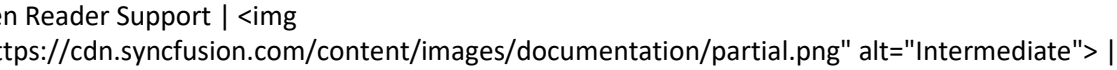
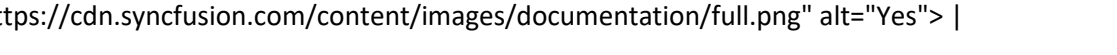
Copyright © 2001 -2024 Syncfusion Inc.

```
}
},
methods: {
loaded: function(args) {
args.smithchart.export('PNG', 'export');
}
}
}
</script>
`
```

Accessibility in Vue Smithchart component

The Smith chart component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Smith chart component is outlined below.

Accessibility Criteria	Compatibility
WCAG 2.2 Support	
Section 508 Support	
Screen Reader Support	
Color Contrast	
Mobile Device Support	
Keyboard Navigation Support	
Accessibility Checker Validation	
Axe-core Accessibility Validation	

```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
```

}

</style>

<img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/not-supported.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Smith chart component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Smith chart component:

- img (role)
- region (role)
- aria-label (attribute)
- aria-hidden (attribute)

Keyboard interaction

The Smith chart component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Smith chart component.

| **Press** | **To do this** |

| --- | --- |

| Tab | Moves the focus to the next element in the Smith chart. |

| Shift + Tab | Moves the focus to the previous element in the Smith chart. |

| Ctrl + P | Prints the Smith chart. |

Ensuring accessibility

The Smith chart component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Smith chart component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Smith chart component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/smith-chart.html" %}

See also

- Accessibility in Syncfusion Vue components

Copyright © 2001 -2024 Syncfusion Inc.

Sparkline

Getting Started with the Vue Sparkline Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLILink to the Video and integrating the Syncfusion Vue Sparkline component.

To get start quickly with Vue Sparkline, you can check on this video:

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of minimum dependencies are required to use the sparkline:

`js

|-- @syncfusion/ej2-vue-charts

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-data

|-- @syncfusion/ej2-pdf-export

|-- @syncfusion/ej2-file-utils

|-- @syncfusion/ej2-compression

|-- @syncfusion/ej2-svg-base

|-- @syncfusion/ej2-sparkline

|-- @syncfusion/ej2-vue-base

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

vue create quickstart

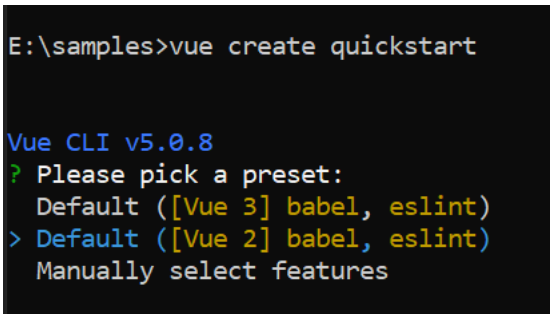
cd quickstart

yarn run serve

Copyright © 2001 -2024 Syncfusion Inc.

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Sparkline component as an example. Install the @syncfusion/ej2-vue-charts package by running the following command:

`bash

npm install @syncfusion/ej2-vue-charts --save

`

or

`bash

yarn add @syncfusion/ej2-vue-charts

`

Add Syncfusion Vue component

Follow the below steps to add the Vue Sparkline component:

1\. First, import and register the Sparkline component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { SparklineComponent } from "@syncfusion/ej2-vue-charts";
export default {
components: {
'ejs-sparkline': SparklineComponent
}
}
</script>
```

2\. In the template section, define the Sparkline component.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-sparkline></ejs-sparkline>
</div>
</template>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

The following example shows a basic Sparkline.

`

<template>

<ejs-sparkline id="sparkline"></ejs-sparkline>

</template>

<script>

import { SparklineComponent } from '@syncfusion/ej2-vue-charts';

export default {

components: {

'ejs-sparkline': SparklineComponent

}

}

</script>

`

Since the data source has not been specified to the sparkline, no shape will be rendered. Only an empty SVG element is appended to the sparkline container.

Module Injection

The sparkline component is segregated into individual feature-wise modules. To use a particular feature, inject its feature module using the provide option on component creation. The module available in sparkline and its descriptions is as follows.

- SparklineTooltip - Inject this provider to use tooltip series.

Copyright © 2001 -2024 Syncfusion Inc.

In the current application, the above basic sparkline is modified to visualize the types of sparkline.

In this application, tooltip feature of the sparkline is used. Now, import the SparklineTooltip module from the sparkline package, and inject it into the sparkline control using the provide option.

```
<template>
<div class="control_wrapper">
<ejs-sparkline id="sparkline"></ejs-sparkline>
</div>
</template>
<script>
import { SparklineComponent, SparklineTooltip } from "@syncfusion/ej2-vue-charts";
export default {
components: {
'ejs-sparkline': SparklineComponent
},
data: function() {
return {
}
},
provide:{
sparkline:[SparklineTooltip]
}
}
</script>
```

Bind data source to Sparkline

The [dataSource] property is used for binding data source to the sparkline. This property takes the collection value as input. For example, the list of objects can be provided as input.

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
    <div>
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xval' yName='yval' :height='height'
:width='width'></ejs-sparkline>
    </div>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import { SparklineComponent } from "@syncfusion/ej2-vue-charts";
export default {
  components: {
    'ejs-sparkline': SparklineComponent
  },
  data: function() {
    return {
    height: '100px',
    width: '70%',
    dataSource: [
            { x: 0, xval: '2005', yval: 20090440 },
            { x: 1, xval: '2006', yval: 20264080 },
            { x: 2, xval: '2007', yval: 20434180 },
            { x: 3, xval: '2008', yval: 21007310 },
            { x: 4, xval: '2009', yval: 21262640 },
            { x: 5, xval: '2010', yval: 21515750 },
            { x: 6, xval: '2011', yval: 21766710 },
            { x: 7, xval: '2012', yval: 22015580 },
            { x: 8, xval: '2013', yval: 22262500 },
            { x: 9, xval: '2014', yval: 22507620 },
        ]
    }
  }
}
</script>
<style>
.spark {
    border: 1px solid rgb(209, 209, 209);
    border-radius: 2px;
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs14" %}

Change the type of Sparkline

You can change the sparkline type by setting the [type] property to [Line], [Column], [WinLoss], [Pie], or [Area]. Here, the sparkline type has been set to [area].

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
    <div>
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xval' yName='yval' :type='type'
:height='height' :width='width'></ejs-sparkline>
        </div>
    </div>
</template>
<script>
import { SparklineComponent } from "@syncfusion/ej2-vue-charts";
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  components: {
    'ejs-sparkline': SparklineComponent
  },
  data: function() {
    return {
    height: '100px',
    width: '70%',
    dataSource: [
            { x: 0, xval: '2005', yval: 20090440 },
            { x: 1, xval: '2006', yval: 20264080 },
            { x: 2, xval: '2007', yval: 20434180 },
            { x: 3, xval: '2008', yval: 21007310 },
            { x: 4, xval: '2009', yval: 21262640 },
            { x: 5, xval: '2010', yval: 21515750 },
            { x: 6, xval: '2011', yval: 21766710 },
            { x: 7, xval: '2012', yval: 22015580 },
            { x: 8, xval: '2013', yval: 22262500 },
            { x: 9, xval: '2014', yval: 22507620 },
        ],
    // Assign the 'area' as type of Sparkline
    type:'Area'
    }
  }
}
</script>
<style>
.spark {
    border: 1px solid rgb(209, 209, 209);
    border-radius: 2px;
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs15" %}

Enable tooltip for Sparkline

The sparkline displays additional information through tooltip when the mouse is hovered over the sparkline. You can enable tooltip by setting the [visible] property to true in [tooltipSettings] and injecting SparklineTooltip module using the provide option.

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
    <div>
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xval' yName='yval' :type='type'
:tooltipSettings='tooltipSettings' :height='height' :width='width'></ejs-
sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { SparklineComponent, SparklineTooltip } from "@syncfusion/ej2-vue-
charts";
export default {
  components: {
    'ejs-sparkline': SparklineComponent
  },
  data: function() {
    return {
    height: '100px',
    width: '70%',
    dataSource: [
            { x: 0, xval: '2005', yval: 20090440 },
            { x: 1, xval: '2006', yval: 20264080 },
            { x: 2, xval: '2007', yval: 20434180 },
            { x: 3, xval: '2008', yval: 21007310 },
            { x: 4, xval: '2009', yval: 21262640 },
            { x: 5, xval: '2010', yval: 21515750 },
            { x: 6, xval: '2011', yval: 21766710 },
            { x: 7, xval: '2012', yval: 22015580 },
            { x: 8, xval: '2013', yval: 22262500 },
            { x: 9, xval: '2014', yval: 22507620 },
        ],
    // Assign the 'area' as type of Sparkline
    type:'Area',
     // To enable the tooltip for Sparkline
    tooltipSettings: {
        visible: true,
        format: '${xval} : ${yval}'
    }
    }
  },
provide:{
    sparkline:[SparklineTooltip]
}
}
</script>
<style>
.spark {
    border: 1px solid rgb(209, 209, 209);
    border-radius: 2px;
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs16" %}

Getting Started with the Vue Sparkline Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Sparkline component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

Copyright © 2001 -2024 Syncfusion Inc.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Sparkline component as an example. To use the Vue Sparkline component in the project, the @syncfusion/ej2-vue-charts package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-charts --save

`

or

`bash

yarn add @syncfusion/ej2-vue-charts

`

Add Syncfusion Vue component

Follow the below steps to add the Vue Sparkline component using Composition API or Options API:

Copyright © 2001 -2024 Syncfusion Inc.

1.First, import and register the Sparkline component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SparklineComponent as EjsSparkline, SparklineTooltip } from
"@syncfusion/ej2-vue-charts";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SparklineComponent, SparklineTooltip } from "@syncfusion/ej2-vue-
charts";
//Component registration
export default {
name: "App",
components: {
"ejs-sparkline": SparklineComponent
}
}
</script>
```

2.In the template section, define the Sparkline component with the dataSource property.

~/SRC/APP.VUE

```
<template>
<ejs-sparkline id="sparkline" :dataSource='dataSource' :height='height'
:xName='xName' :yName='yName' :type='type'
:tooltipSettings='tooltipSettings'></ejs-sparkline>
</template>
```

3.Declare the values for the dataSource property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const dataSource = [
{ x: 0, xval: '2005', yval: 20090440 },
{ x: 1, xval: '2006', yval: 20264080 },
{ x: 2, xval: '2007', yval: 20434180 },
{ x: 3, xval: '2008', yval: 21007310 },
{ x: 4, xval: '2009', yval: 21262640 },
{ x: 5, xval: '2010', yval: 21515750 },
{ x: 6, xval: '2011', yval: 21766710 },
{ x: 7, xval: '2012', yval: 22015580 },
{ x: 8, xval: '2013', yval: 22262500 },
{ x: 9, xval: '2014', yval: 22507620 }
];
const type = 'Area';
const xName = 'xval';
const yName = 'yval';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
const height = '200px';
const tooltipSettings = {
visible: true,
format: '${xval} : ${yval}'
};
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
dataSource: [
{ x: 0, xval: '2005', yval: 20090440 },
{ x: 1, xval: '2006', yval: 20264080 },
{ x: 2, xval: '2007', yval: 20434180 },
{ x: 3, xval: '2008', yval: 21007310 },
{ x: 4, xval: '2009', yval: 21262640 },
{ x: 5, xval: '2010', yval: 21515750 },
{ x: 6, xval: '2011', yval: 21766710 },
{ x: 7, xval: '2012', yval: 22015580 },
{ x: 8, xval: '2013', yval: 22262500 },
{ x: 9, xval: '2014', yval: 22507620 }
],
type:'Area',
xName:'xval',
yName:'yval',
height:'200px',
tooltipSettings: {
visible: true,
format: '${xval} : ${yval}'
}
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-sparkline id="sparkline" :dataSource='dataSource' :height='height'
:xName='xName' :yName='yName' :type='type'
:tooltipSettings='tooltipSettings'></ejs-sparkline>
</template>
<script setup>
import { provide } from 'vue';
import { SparklineComponent as EjsSparkline, SparklineTooltip } from
"@syncfusion/ej2-vue-charts";
const dataSource = [
{ x: 0, xval: '2005', yval: 20090440 },
{ x: 1, xval: '2006', yval: 20264080 },
{ x: 2, xval: '2007', yval: 20434180 },
{ x: 3, xval: '2008', yval: 21007310 },
{ x: 4, xval: '2009', yval: 21262640 },
{ x: 5, xval: '2010', yval: 21515750 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ x: 6, xval: '2011', yval: 21766710 },
{ x: 7, xval: '2012', yval: 22015580 },
{ x: 8, xval: '2013', yval: 22262500 },
{ x: 9, xval: '2014', yval: 22507620 }
];
const type = 'Area';
const xName = 'xval';
const yName = 'yval';
const height = '200px';
const tooltipSettings = {
visible: true,
format: '${xval} : ${yval}'
};
const sparkline = [SparklineTooltip];
provide('sparkline', sparkline);
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-sparkline id="sparkline" :dataSource='dataSource' :height='height'
:xName='xName' :yName='yName' :type='type'
:tooltipSettings='tooltipSettings'></ejs-sparkline>
</template>
<script>
import { SparklineComponent, SparklineTooltip } from "@syncfusion/ej2-vue-
charts";
//Component registration
export default {
name: "App",
components: {
"ejs-sparkline": SparklineComponent
},
data() {
return {
dataSource: [
{ x: 0, xval: '2005', yval: 20090440 },
{ x: 1, xval: '2006', yval: 20264080 },
{ x: 2, xval: '2007', yval: 20434180 },
{ x: 3, xval: '2008', yval: 21007310 },
{ x: 4, xval: '2009', yval: 21262640 },
{ x: 5, xval: '2010', yval: 21515750 },
{ x: 6, xval: '2011', yval: 21766710 },
{ x: 7, xval: '2012', yval: 22015580 },
{ x: 8, xval: '2013', yval: 22262500 },
{ x: 9, xval: '2014', yval: 22507620 }
],
type:'Area',
xName:'xval',
yName:'yval',
height:'200px',
tooltipSettings: {
visible: true,
format: '${xval} : ${yval}'
}
};
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
provide: {
sparkline:[SparklineTooltip]
},
};
</script>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

Sample: vue-3-sparkline-charts-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Sparkline dimensions in Vue Sparkline component

Size for container

Sparkline can be rendered to its container size. You can set the size through inline or CSS as shown in the following code.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xval' yName='yval' :height='height'
:width='width'></ejs-sparkline>
    </div>
    </div>
```

```
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '350px',
    width: '650px',
    dataSource: [
            { x: 0, xval: '2005', yval: 20090440 },
            { x: 1, xval: '2006', yval: 20264080 },
            { x: 2, xval: '2007', yval: 20434180 },
            { x: 3, xval: '2008', yval: 21007310 },
            { x: 4, xval: '2009', yval: 21262640 },
            { x: 5, xval: '2010', yval: 21515750 },
            { x: 6, xval: '2011', yval: 21766710 },
            { x: 7, xval: '2012', yval: 22015580 },
            { x: 8, xval: '2013', yval: 22262500 },
            { x: 9, xval: '2014', yval: 22507620 },
        ]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs24" %}

<!-- markdownlint-disable MD036 -->

Size for sparkline

<!-- markdownlint-disable MD036 -->

You can also set the size for sparkline directly using the width and height properties.

In pixel

You can set the size for sparkline in pixel as demonstrated in the following code.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xval' yName='yval' :height='height'
:width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '150px',
    width: '350px',
    dataSource: [
            { x: 0, xval: '2005', yval: 20090440 },
            { x: 1, xval: '2006', yval: 20264080 },
            { x: 2, xval: '2007', yval: 20434180 },
            { x: 3, xval: '2008', yval: 21007310 },
            { x: 4, xval: '2009', yval: 21262640 },
            { x: 5, xval: '2010', yval: 21515750 },
            { x: 6, xval: '2011', yval: 21766710 },
            { x: 7, xval: '2012', yval: 22015580 },
            { x: 8, xval: '2013', yval: 22262500 },
            { x: 9, xval: '2014', yval: 22507620 },
        ]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs25" %}

In percentage

By setting values in percentage, sparkline gets its dimension with respect to its container. For example, when the height is set to ‘50%’, sparkline is rendered to half of its container height.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xval' yName='yval' :height='height'
:width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '50%',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    width: '80%',
    dataSource: [
            { x: 0, xval: '2005', yval: 20090440 },
            { x: 1, xval: '2006', yval: 20264080 },
            { x: 2, xval: '2007', yval: 20434180 },
            { x: 3, xval: '2008', yval: 21007310 },
            { x: 4, xval: '2009', yval: 21262640 },
            { x: 5, xval: '2010', yval: 21515750 },
            { x: 6, xval: '2011', yval: 21766710 },
            { x: 7, xval: '2012', yval: 22015580 },
            { x: 8, xval: '2013', yval: 22262500 },
            { x: 9, xval: '2014', yval: 22507620 },
        ]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs26" %}

Sparkline types in Vue Sparkline component

Different types of shapes can be used to represent the sparkline. You can change the sparkline type by setting the type property. Sparkline supports the following types:

- Line
- Column
- Win-Loss
- Pie
- Area

The following code sample shows different types of sparklines.

<!-- markdownlint-disable MD036 -->

Line

The [Line] type is used to render the sparkline series as line.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xval' yName='yval' :height='height'
:type='type' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
```

```
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '100px',
    width: '70%',
    type:'Line',
    dataSource: [
            { x: 0, xval: '2005', yval: 20090440 },
            { x: 1, xval: '2006', yval: 20264080 },
            { x: 2, xval: '2007', yval: 20434180 },
            { x: 3, xval: '2008', yval: 21007310 },
            { x: 4, xval: '2009', yval: 21262640 },
            { x: 5, xval: '2010', yval: 21515750 },
            { x: 6, xval: '2011', yval: 21766710 },
            { x: 7, xval: '2012', yval: 22015580 },
            { x: 8, xval: '2013', yval: 22262500 },
            { x: 9, xval: '2014', yval: 22507620 },
        ]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs27" %}

Column

The [Column] type is used to render the sparkline series as column.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xval' yName='yval' :type='type'
:height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    height: '150px',
     width: '200px',
     type:'Column',
     dataSource: [
             { x: 0, xval: '2005', yval: 20090440 },
             { x: 1, xval: '2006', yval: 20264080 },
             { x: 2, xval: '2007', yval: 20434180 },
             { x: 3, xval: '2008', yval: 21007310 },
             { x: 4, xval: '2009', yval: 21262640 },
             { x: 5, xval: '2010', yval: 21515750 },
             { x: 6, xval: '2011', yval: 21766710 },
             { x: 7, xval: '2012', yval: 22015580 },
             { x: 8, xval: '2013', yval: 22262500 },
             { x: 9, xval: '2014', yval: 22507620 },
         ]
     }
   }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs28" %}

Pie

The [Pie] type is used to render the sparkline series as pie.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xval' yName='yval' :type='type'
:height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '200px',
    width: '70%',
    type:'Pie',
    dataSource: [
             { x: 0, xval: '2005', yval: 20090440 },
             { x: 1, xval: '2006', yval: 20264080 },
             { x: 2, xval: '2007', yval: 20434180 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            { x: 3, xval: '2008', yval: 21007310 },
            { x: 4, xval: '2009', yval: 21262640 },
            { x: 5, xval: '2010', yval: 21515750 },
            { x: 6, xval: '2011', yval: 21766710 },
            { x: 7, xval: '2012', yval: 22015580 },
            { x: 8, xval: '2013', yval: 22262500 },
            { x: 9, xval: '2014', yval: 22507620 },
        ]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs29" %}

Win Loss

The [WinLoss] type is used to render the sparkline series as Win Loss.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xval' yName='yval' :type='type'
:height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '100px',
    width: '70%',
    type:'WinLoss',
    dataSource: [
            { x: 0, xval: '2005', yval: 20090440 },
            { x: 1, xval: '2006', yval: 20264080 },
            { x: 2, xval: '2007', yval: 20434180 },
            { x: 3, xval: '2008', yval: 21007310 },
            { x: 4, xval: '2009', yval: 21262640 },
            { x: 5, xval: '2010', yval: 21515750 },
            { x: 6, xval: '2011', yval: 21766710 },
            { x: 7, xval: '2012', yval: 22015580 },
            { x: 8, xval: '2013', yval: 22262500 },
            { x: 9, xval: '2014', yval: 22507620 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        ]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs30" %}

Area

The [Area] type is used to render the sparkline series as area.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xval' yName='yval' :type='type'
:height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '100px',
    width: '70%',
    type:'Area',
    dataSource: [
            { x: 0, xval: '2005', yval: 20090440 },
            { x: 1, xval: '2006', yval: 20264080 },
            { x: 2, xval: '2007', yval: 20434180 },
            { x: 3, xval: '2008', yval: 21007310 },
            { x: 4, xval: '2009', yval: 21262640 },
            { x: 5, xval: '2010', yval: 21515750 },
            { x: 6, xval: '2011', yval: 21766710 },
            { x: 7, xval: '2012', yval: 22015580 },
            { x: 8, xval: '2013', yval: 22262500 },
            { x: 9, xval: '2014', yval: 22507620 },
        ]
    }
  }
}
</script>
<style>
.spark {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs31" %}

Axis customization in Vue Sparkline component

You can customize axis value types and min and max values of the sparkline.

Change value type of the sparkline

You can change the sparkline value type by setting the [valueType] property to [Numeric], [Category], or [DateTime].

<!-- markdownlint-disable MD036 -->

DateTime

You can assign date-time values to the sparkline by setting the [valueType] property to [DateTime].

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xDate' yName='yval' :type='type'
:valueType='valueType' :height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '150px',
    width: '130px',
    // Assign the "Column" as type of the sparkline
    type:'Column',
    // Assign the "DateTime" as value type to the sparkline
    valueType: 'DateTime',
    dataSource: [
            { xDate: new Date(2018, 0, 1), x: 0, yval: 4 },
            { xDate: new Date(2018, 0, 2), x: 1, yval: 4.5 },
            { xDate: new Date(2018, 0, 3), x: 2, yval: 8 },
            { xDate: new Date(2018, 0, 4), x: 3, yval: 7 },
            { xDate: new Date(2018, 0, 5), x: 4, yval: 6 },
            { xDate: new Date(2018, 0, 8), x: 5, yval: 8 },
            { xDate: new Date(2018, 0, 9), x: 6, yval: 8 },
            { xDate: new Date(2018, 0, 10), x: 7, yval: 6.5 },
            { xDate: new Date(2018, 0, 11), x: 8, yval: 4 },
            { xDate: new Date(2018, 0, 12), x: 9, yval: 5.5 }
        ]
    }
```

```
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs5" %}

<!-- markdownlint-disable MD036 -->

Category

You can assign category values to the sparkline by setting [valueType] to [Category].

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' xName='xval' yName='yval' :height='height'
:type='type' :valueType='valueType' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '150px',
    width: '130px',
    // Assign the 'Column' as type of the sparkline
    type:'Column',
    // Assign the  "Category" as value type of the sparkline
    valueType: 'Category',
    dataSource: [
            { x: 0, xval: 'Robert', yval: 60 },
            { x: 1, xval: 'Andrew', yval: 65 },
            { x: 2, xval: 'Suyama', yval: 70 },
            { x: 3, xval: 'Michael', yval: 80 },
            { x: 4, xval: 'Janet', yval: 55 },
            { x: 5, xval: 'Davolio', yval: 90 },
            { x: 6, xval: 'Fuller', yval: 75 },
            { x: 7, xval: 'Nancy', yval: 85 }
        ]
    }
  }
}
</script>
<style>
.spark {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs6" %}

Numeric

You can assign numeric values to the sparkline by setting the [valueType] to [Numeric].

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource'  :type='type' :valueType='valueType' xName='x'
yName='yval' :height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '150px',
    width: '130px',
    // Assign the 'Column' as type of the sparkline
    type:'Column',
    // Assign the 'Numeric' as value type of the sparkline
    valueType: 'Numeric',
    dataSource: [
            { x: 1, xval: 2010, yval: 190 },
            { x: 2, xval: 2011, yval: 165 },
            { x: 3, xval: 2012, yval: 158 },
            { x: 4, xval: 2013, yval: 175 },
            { x: 5, xval: 2014, yval: 200 },
            { x: 6, xval: 2015, yval: 180 },
            { x: 7, xval: 2016, yval: 210 }
        ]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs7" %}

Copyright © 2001 -2024 Syncfusion Inc.

<!-- markdownlint-disable MD036 -->

Change min and max values of axis

You can change the min and max values of x-axis by setting the [minX] and [maxX] values to the [axisSettings] property. You can also change the min and max values of y-axis by setting the [minY] and [maxY] values to the [axisSettings] property.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :type= 'type' :axisSettings='axisSettings'
:valueType='valueType' xName='x' yName='yval' :height='height'
:width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '150px',
    width: '130px',
    axisSettings: {
        minY: 0, maxY: 150
    },
    // Assign the 'Column' as type of the sparkline
    type:'Column',
    // Assign the 'Numeric' as value type of the sparkline
    valueType: 'Numeric',
    dataSource: [
            { x: 0, yval: 50 },
            { x: 1, yval: 30 },
            { x: 2, yval: 20 },
            { x: 3, yval: 30 },
            { x: 4, yval: 50 },
            { x: 5, yval: 40 },
            { x: 6, yval: 20 },
            { x: 7, yval: 10 },
            { x: 8, yval: 30 },
            { x: 9, yval: 10 },
            { x: 10, yval: 40 }
        ]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs8" %}

Change value of axis

You can set horizontal axis line value of the sparkline by setting [value] to the [axisSettings] property. The following code example shows this.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :type='type' :valueType= 'valueType'
:axisSettings='axisSettings' xName='x' yName='yval' :height='height'
:width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '150px',
    width: '130px',
   dataSource: [
            { x: 0, yval: 50 },
            { x: 1, yval: 30 },
            { x: 2, yval: 20 },
            { x: 3, yval: 30 },
            { x: 4, yval: 50 },
            { x: 5, yval: 40 },
            { x: 6, yval: 20 },
            { x: 7, yval: 10 },
            { x: 8, yval: 30 },
            { x: 9, yval: 10 },
            { x: 10, yval: 40 }
        ],
    // Set the value to horizontal axis of sparkline
    axisSettings: {
          value: 25
    },
    // Assign the 'Column' as type of the sparkline
    type:'Column',
    // Assign the 'Numeric' as value type of the sparkline
    valueType: 'Numeric'
    }
  }
}
</script>
<style>
```

```
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs9" %}

Axis line customization

Axis of the sparkline can be collapsed using the [visible] property in [lineSettings]; this is not applicable for win-loss. You can customize the [color], [width], [opacity], and [dashArray] of axis line.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :axisSettings='axisSettings' :valueType='valueType'
xName='x' yName='yval' :height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '150px',
    width: '130px',
    dataSource: [
            { x: 0, yval: 50 },
            { x: 1, yval: 30 },
            { x: 2, yval: 20 },
            { x: 3, yval: 30 },
            { x: 4, yval: 50 },
            { x: 5, yval: 40 },
            { x: 6, yval: 20 },
            { x: 7, yval: 10 },
            { x: 8, yval: 30 },
            { x: 9, yval: 10 },
            { x: 10, yval: 40 }
        ],
    // Set the value to horizontal axis
    axisSettings: {
    // To configure axis line settings
         lineSettings: {
            visible: true,
            color: "#ff14ae",
            dashArray:5
            }
    },
    // Assign the value type as 'Numeric' to the sparkline
    valueType: 'Numeric'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs10" %}

Special points customization in Vue Sparkline component

You can customize the points by initializing the point colors. The customization options allows to differentiate the [start], [end], [positive], [negative], and [low] points. This customization is only applicable for line, column, and area type sparklines.

<!-- markdownlint-disable MD036 -->

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource':highPointColor ='highPointColor' :lowPointColor=
'lowPointColor' :startPointColor ='startPointColor'
:endPointColor='endPointColor' :negativePointColor='negativePointColor'
:type='type' :valueType='valueType' xName='xval' yName='yval'
:height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '150px',
    width: '130px',
    dataSource: [
            { x: 0, xval: 'AUDI', yval: 1 },
            { x: 1, xval: 'BMW', yval: 5 },
            { x: 2, xval: 'BUICK', yval: -1 },
            { x: 3, xval: 'CETROEN', yval: -6 },
            { x: 4, xval: 'CHEVROLET', yval: 0 },
            { x: 5, xval: 'FIAT', yval: 1 },
            { x: 6, xval: 'FORD', yval: -2 },
            { x: 7, xval: 'HONDA', yval: 7 },
            { x: 8, xval: 'HYUNDAI', yval: -9 },
            { x: 9, xval: 'JEEP', yval: 0 },
            { x: 10, xval: 'KIA', yval: -10 },
            { x: 11, xval: 'MAZDA', yval: 3 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        ],
    type:'Column',
    valueType: 'Category',
    highPointColor:'blue',
    lowPointColor:'orange',
    startPointColor:'green',
    endPointColor:'green',
    negativePointColor:'red'
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs32" %}

Tie point color

Tie point color is used to configure the win-loss series type sparkline's y-value point color. The following code sample shows the tie point color of sparkline series.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :tiePointColor='tiePointColor' :type='type'
:valueType='valueType' :height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '150px',
    width: '130px',
    dataSource: [12, 15, -10, 13, 15, 6, -12, 17, 13, 0, 8, -10],
    // Assign the 'WinLoss' as type of Sparkline
    type:'WinLoss',
    // Assign "Numeric" as the value type of the sparkline
    valueType: 'Numeric',
    tiePointColor:'blue'
    }
  }
}
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs33" %}

Range band in Vue Sparkline component

This section explains how to customize the sparkline with multiple range bands.

Range band customization

The range band feature is used to highlight a particular range along with the y-axis using the [startRange] and [endRange] properties. You can also customize the [color] and [opacity] of the range band.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :rangeBandSettings ='rangeBandSettings' fill=
'#0d3c9b' :lineWidth='lineWidth' :height='height' :width='width'></ejs-
sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '150px',
    width: '150px',
    lineWidth: 2,
    dataSource:   [0, 6, 4, 1, 3, 2, 5],
    // To configure range band settings
    rangeBandSettings: [
            {
                startRange: 1,
                endRange: 3,
                color: '#bfd4fc',
                opacity:0.4
            }
    ]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
```

```
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs22" %}

Multiple range band customization

You can define multiple range bands to a sparkline as shown in the following code sample.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' fill= '#0d3c9b'
:rangeBandSettings='rangeBandSettings' :lineWidth='lineWidth'
:height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '150px',
    width: '150px',
    lineWidth: 2,
    dataSource:   [0, 6, 4, 1, 3, 2, 5],
    rangeBandSettings: [
            {
                startRange: 1,
                endRange: 2,
                color: '#bfd4fc',
                opacity:0.4
            },
            {
                startRange: 4,
                endRange: 5,
                color: 'red',
                opacity:0.4
            }
        ]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs23" %}

Marker in Vue Sparkline component

This section explains how to add markers to the sparklines.

Adding marker to the sparkline

To add marker to the sparkline, specify the `visible` of `markerSettings` as following values. The `visible` will accept multiple values too.

- All - Enables markers for all points.
- Start - Enables marker for the start point.
- End - Enables marker for the end point.
- High - Enables marker for the high point.
- Low - Enables marker for the low point.
- Negative - Enables markers for the negative points.

The following code example shows enabling markers for all points.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :axisSettings='axisSettings'
:markerSettings='markerSettings' :height='height' :width='width'></ejs-
sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '200px',
    width: '350px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 7, minY: -1
    }
    dataSource:   [0, 6, 4, 1, 3, 2, 5],
    // To enable markers for all points
    markerSettings: {
        visible: ['All']
    }
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
```

```
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs19" %}

Adding marker to special point

In sparkline, markers can be enabled for particular points such as the start, end, low, high, or negative. The following code examples shows enabling markers for the high and low points.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :lowPointColor='lowPointColor'
:markerSettings='markerSettings' :highPointColor='highPointColor'
:axisSettings='axisSettings' :height='height' :width='width'></ejs-
sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '200px',
    width: '350px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 7, minY: -1
    }
    dataSource:   [3, 6, 4, 1, 3, 2, 5],
    // To enable marker for high and low points with color customization
    highPointColor: 'blue',
    lowPointColor: 'red',
    markerSettings: {
        visible: ['High', 'Low']
    }
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs20" %}

Copyright © 2001 -2024 Syncfusion Inc.

Customizing markers

Sparkline markers can be customized in terms of fill color, border color, width, opacity, and size. The following code example shows customizing marker's fill, border, and size.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :markerSettings='markerSettings' fill=
'blue':axisSettings='axisSettings' :height='height' :width='width'></ejs-
sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '200px',
    width: '350px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 7, minY: -1
    }
    dataSource:   [3, 6, 4, 1, 3, 2, 5],
    // To enable marker for high and low points with color customization
    markerSettings: {
        visible: ['All'],
        size: 5, fill: 'white', border: { color: 'blue', width: 2}
    }
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs21" %}

Data labels in Vue Sparkline component

Data labels are used to display values of data points to improve the readability.

Enable data label

To enable data label for sparkline series, provide `visible` of the `dataLabelSettings` as following possible values:

- All - Enables data label of all points.

Copyright © 2001 -2024 Syncfusion Inc.

- Start - Enables data label of the start point.
- End - Enables data label of the end point.
- High - Enables data label of the high point.
- Low - Enables data label of the low point.
- Negative - Enables data labels of the negative points.

The following example shows enabling sparkline data label for all points.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :dataLabelSettings='dataLabelSettings' fill= 'blue'
:axisSettings='axisSettings' :height='height' :width='width'></ejs-
sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '200px',
    width: '350px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 7, minY: -1
    },
    dataSource:   [0, 6, 4, 1, 3, 2, 5],
    // To enable data label for all points
    dataLabelSettings: {
        visible: ['All']
    },
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs11" %}

Customize data label

Data labels can be customized using the fill, border, opacity, and textStyle. The following code example shows customizing data label border, text color, and fill color.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:axisSettings='axisSettings' fill= 'blue'
:dataLabelSettings='dataLabelSettings' :dataSource='dataSource'
:height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '200px',
    width: '350px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 8, minY: -1
    },
    dataSource:   [0, 6, 4, 1, 3, 2, 5],
    // To customize data label fill, border, and text color
    dataLabelSettings: {
        visible: ['All'],
        border: { color: 'blue', width: 1},
        fill: 'blue', opacity: 0.4,
        textStyle: {
            color: 'white'
        }
    }
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs12" %}

Format data label text

The text of data labels can be formatted using the format API in the sparkline dataLabelSettings. By default, data label shows the y-value of point. The following code example shows how to display x and y-values for points.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
```

```
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :axisSettings='axisSettings'
:dataLabelSettings='dataLabelSettings' fill= 'blue'valueType= 'Category'
xName='x' yName='y' :height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
     height: '200px',
    width: '500px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 8, minY: -1
    },
    dataSource:   [{x: 'Mon', y: 3 },{x: 'Tue', y: 5 },{x: 'Wed', y: 2 },{x:
'Thu', y: 4 },{x: 'Fri', y: 6 },],
    // To enable data label for all points
    dataLabelSettings: {
        format: '${x} : ${y}'
        visible: ['All'],
        border: { color: 'blue', width: 1},
        fill: 'blue', opacity: 0.4,
        textStyle: {
            color: 'white'
        }
    }
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs13" %}

User interaction in Vue Sparkline component

Sparkline has two user interaction features: tooltip and tracker line.

Tooltip

The sparkline provides options to display details about values of data points through tooltips when hovering the mouse over data point. To use tooltip in sparkline, inject the `SparklineTooltip` module to sparkline using the inject method.

The following code example shows enabling tooltip for sparkline with format.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :axisSettings='axisSettings' :valueType='valueType'
fill= 'blue' xName='x' yName='y' :tooltipSettings='tooltipSettings'
:height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin,SparklineTooltip } from "@syncfusion/ej2-vue-
charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '200px',
    width: '500px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 8, minY: -1
    },
    valueType: 'Category',
    dataSource:   [{x: 'Mon', y: 3 },{x: 'Tue', y: 5 },{x: 'Wed', y: 2 },{x:
'Thu', y: 4 },{x: 'Fri', y: 6 }],
    // To enable tooltip for sparkline
    tooltipSettings: {
        visible: true,
        format: '${x} : ${y}'
    }
    }
  },
provide:{
    sparkline:[SparklineTooltip]
}
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs34" %}

Tooltip customization

The fill color, text styles, format, and border of the tooltip can be customized. The following code example shows customization of tooltip's fill color and text style.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
```

```
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :axisSettings='axisSettings' :valueType='valueType'
fill= 'blue' xName='x' yName='y' :tooltipSettings='tooltipSettings'
:height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin,SparklineTooltip } from "@syncfusion/ej2-vue-
charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '200px',
    width: '500px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 8, minY: -1
    },
    valueType: 'Category',
    dataSource:   [{x: 'Mon', y: 3 },{x: 'Tue', y: 5 },{x: 'Wed', y: 2 },{x:
'Thu', y: 4 },{x: 'Fri', y: 6 }],
    // To enable tooltip for sparkline
    tooltipSettings: {
        visible: true,
        format: '${x} : ${y}',
        fill: '#033e96',
        textStyle: {
            color: 'white'
        },
    }
    }
  },
provide:{
    sparkline:[SparklineTooltip]
}
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs35" %}

Tooltip template

Sparkline tooltip has template support. By using tooltip template, you can customize tooltips. The following code example shows more customization options provided to `sparktooltip` css class that is used in tooltip template div. Using this template, images also can be added to tooltip.

`

.sparktooltip {

border-radius: 5px;

background: #008cff;

color: #FFFFFF !important;

font-size: 16px;

font-style: italic;

padding: 8px;

}

`

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :axisSettings='axisSettings'  valueType= 'Category'
fill= '#033e96' xName='x' yName='y' :tooltipSettings='tooltipSettings'
:height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin,SparklineTooltip } from "@syncfusion/ej2-vue-
charts";
var contentTemplate = function() {
  return { template: contentVue };
};
var contentVue = Vue.component("contentTemplate", {
  template: '<div style="border-radius: 5px; background: #008cff;color:
#FFFFFF !important;font-size: 16px;font-style: italic;padding:
8px;">{{data.x}} : {{data.y}}</div>',
  data() {
    return {
      data: {}
    };
  }
});
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '200px',
    width: '500px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 8, minY: -1
    },
    dataSource:   [{x: 'Mon', y: 3 },{x: 'Tue', y: 5 },{x: 'Wed', y: 2 },{x:
'Thu', y: 4 },{x: 'Fri', y: 6 },],
    // To enable tooltip template for sparkline with fill color, border
radius and padding customization
    tooltipSettings: {
```

```
        visible: true,
        template: contentTemplate
    }
    }
  },
provide:{
    sparkline:[SparklineTooltip]
}
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs36" %}

Track line

The track line tracks data points that are closer to the mouse position or touch contact.

To enable track lines for sparkline, specify the `visible` option of `trackLineSettings` to true. Based on theme, tracker color will be changed. The default value of tracker color is black.

To use track line in sparkline, inject the `SparklineTooltip` module to sparkline using the inject method.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' fill= '#033e96' :axisSettings='axisSettings'
:tooltipSettings='tooltipSettings' :height='height' :width='width'></ejs-
sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin,SparklineTooltip } from "@syncfusion/ej2-vue-
charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '200px',
    width: '500px',
    axisSettings: {
        minX: -1, maxX: 46, maxY: 10, minY: -1
    },
    dataSource:   [5, 3, 4, 6, 8, 7, 9, 1, 3, 5, 3, 4, 6, 8, 7, 9, 1, 3, 5,
2, 4, 6, 7, 9, 5, 8, 3, 6, 1, 7, 4, 2, 5, 2, 4, 6, 7, 9, 5, 8, 3, 6, 1, 7,
4, 2],
    // To enable tooltip template for sparkline with fill color, border
radius and padding customization
```

```
    tooltipSettings: {
        trackLineSettings: {
            visible: true,
            color: '#033e96',
            width: 1
        }
    }
    }
  },
provide:{
    sparkline:[SparklineTooltip]
}
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs37" %}

Appearance in Vue Sparkline component

The appearance of the sparkline can be customized using margin, container Area border, and container Area background.

Sparkline border

The container Area border of the sparkline is used to render border to cover sparkline area.

The following code example shows the sparkline with overall border.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :border='border' fill= '#b2cfff'
:containerArea='containerArea' :type='type' :height='height'
:width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    // To render border around the sparkline
    containerArea: {
        border: { color: '#033e96', width: 2 }
    },
    border: { color: '#033e96', width: 1 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    height: '200px',
    width: '350px',
    type: 'Area',
    fill: '#b2cfff'
    dataSource: [3, 6, 4, 1, 3, 2, 5]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs1" %}

Sparkline padding

Padding is used to specify padding value between container and sparkline. By default, padding value of the sparkline is 5. Sparkline padding values are specified by the left, right, top, and bottom.

The following code example shows the sparkline with overall padding is set to 20.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :border='border' :containerArea='containerArea'
:padding='padding' fill= '#b2cfff' :type='type' :height='height'
:width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    containerArea: {
        border: { color: '#033e96', width: 2 }
    },
    // To render sparkline with padding
    padding: { left: 20, right: 20, bottom: 20, top: 20},
    border: { color: '#033e96', width: 1 },
    height: '200px',
    width: '350px',
    type: 'Area',
    dataSource: [3, 6, 4, 1, 3, 2, 5]
    }
  }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs2" %}

Sparkline area customization

The background color of the sparkline area can be customized using the containerArea background color. By default, the sparkline background color is transparent.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center" :padding='padding'
:border='border' fill= '#b2cfff' :containerArea='containerArea'
:dataSource='dataSource' :type='type' :height='height' :width='width'></ejs-
sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    containerArea: {
        border: { color: '#033e96', width: 2 },
        // To render sparkline with background
        background: '#eff1f4',
    },
    padding: { left: 20, right: 20, bottom: 20, top: 20},
    border: { color: '#033e96', width: 2 },
    height: '200px',
    width: '350px',
    type: 'Area',
    dataSource: [3, 6, 4, 1, 3, 2, 5]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs3" %}

Sparkline theme

Datalabel and track line colors of the sparkline will be changed based on theme. For example, for dark theme, the color of datalabel and track line should be white; for light theme, their value should be black. The possible values for sparkline theme are Material, Fabric, Bootstrap, and Highcontrast.

The following code example shows the color for datalabel and track line is set to white for dark theme.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :border='border' fill= '#007dd1' lineWidth= 3
:axisSettings='axisSettings' :dataLabelSettings='dataLabelSettings' theme=
'Highcontrast' :type='type' :tooltipSettings='tooltipSettings'
:height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin,SparklineTooltip } from "@syncfusion/ej2-vue-
charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    dataLabelSettings: { visible: ['All']},
    tooltipSettings: {
        trackLineSettings: { visible: true }
    },
    axisSettings: {
        minX: -1, maxX: 7
    },
    border: { color: 'transparent', width: 2 },
    height: '200px',
    width: '350px',
    type: 'Line',
    dataSource: [3, 6, 4, 1, 3, 2, 5]
    }
  },
provide:{
    sparkline:[SparklineTooltip]
}
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs4" %}

Localization in Vue Sparkline component

The sparkline control supports localization. The default culture for localization is en-US. You can change the culture using the setCulture method.

Tooltip format

Sparkline tooltip supports localization. The following code sample shows tooltip text with currency format based on culture.

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :containerArea='containerArea' :border='border'
fill= '#b2cfff' :padding='padding' lineWidth= 3 format= 'c0'
useGroupingSeparator= 'true' :type='type' :tooltipSettings='tooltipSettings'
:height='height' :width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin,SparklineTooltip } from "@syncfusion/ej2-vue-
charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
   containerArea: {
        border: { color: '#033e96', width: 2 }
    },
    tooltipSettings: {
        visible: true
    },
    padding: { left: 20, right: 20, bottom: 20, top: 20},
    border: { color: '#033e96', width: 2 },
    height: '200px',
    width: '350px',
    type: 'Area',
    dataSource: [30000, 60000, 40000, 10000, 30000, 20000, 50000]
    }
  },
provide:{
    sparkline:[SparklineTooltip]
}
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs17" %}

Rtl

If you set the enableRtl property to true, then the sparkline will be rendered from rigt-to-left direction.

The following example shows the sparkline is render from "Right-to-left".

APP.VUE

```
<template>
    <div class="control_wrapper">
    <div class="spark">
        <ejs-sparkline id="sparkline" align="center"
:dataSource='dataSource' :enableRtl='enableRtl' :height='height'
:width='width'></ejs-sparkline>
    </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { SparklinePlugin,SparklineTooltip } from "@syncfusion/ej2-vue-
charts";
Vue.use(SparklinePlugin);
export default {
  data: function() {
    return {
    height: '150px',
    width: '150px',
    enableRtl: true
    dataSource: [30000, 60000, 40000, 10000, 30000, 20000, 50000]
    }
  }
}
</script>
<style>
.spark {
    width: 100%;
    height: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/sparkline/getting-started-cs18" %}

Accessibility in Vue Sparkline component

The Sparkline component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Sparkline component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Sparkline component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Sparkline component:

- img (role)
- aria-label (attribute)
- aria-hidden (attribute)

Keyboard interaction

The Sparkline component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Sparkline component.

Copyright © 2001 -2024 Syncfusion Inc.

| **Press** | **To do this** |

| --- | --- |

| Ctrl + P | Prints the Sparkline. |

Ensuring accessibility

The Sparkline component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Sparkline component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Sparkline component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/spark-line-chart.html" %}

See also

- Accessibility in Syncfusion Vue components

SpeedDial

Getting Started with the Vue Speed dial Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Speed dial component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The list of dependencies required to use the Speed Dial component in your application is given as follows:

`js

|-- @syncfusion/ej2-vue-buttons

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-buttons

|-- @syncfusion/ej2-vue-base

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

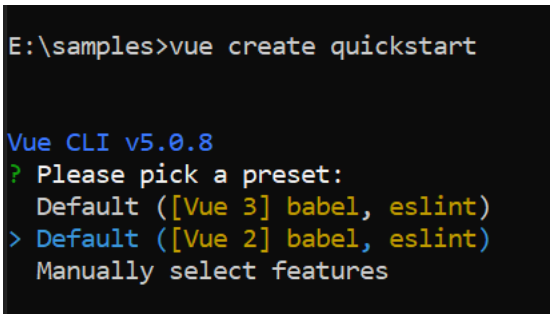
npm run serve

Copyright © 2001 -2024 Syncfusion Inc.

```
or

```bash
yarn global add @vue/cli
vue create quickstart
cd quickstart
yarn run serve
```

When creating a new project, choose the option `Default ([Vue 2] babel, eslint)` from the menu.



Once the `quickstart` project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Speed dial component as an example. Install the `@syncfusion/ej2-vue-buttons` package by running the following command:

```bash
npm install @syncfusion/ej2-vue-buttons --save
```

or

```bash
yarn add @syncfusion/ej2-vue-buttons
```

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

Copyright © 2001 -2024 Syncfusion Inc.
```

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Speed dial component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Speed dial component using Composition API or Options API:

1\. First, import and register the Speed dial component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SpeedDialComponent as EjsSpeeddial } from "@syncfusion/ej2-vue-buttons";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SpeedDialComponent } from "@syncfusion/ej2-vue-buttons";
export default {
components: {
'ejs-speeddial': SpeedDialComponent
}
}
</script>
```

2\. In the template section define the Speed dial component with, content, target and items property.

~/SRC/APP.VUE

```
<template>
<div>
<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
<ejs-speeddial id='speeddial'  content='Edit' target='#targetElement' :items='items'></ejs-speeddial>
</div>
</template>
```

3\. Declare the value for the items property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
const items = [
{
text:'Cut'
},
{
text:'Copy'
},
{
text:'Paste'
}
];
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
items: [
{
text:'Cut'
},
{
text:'Copy'
},
{
text:'Paste'
}
]
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial'  content='Edit'
target='#targetElement' :items='items'></ejs-speeddial>
    </div>
</template>
<script setup>
import { SpeedDialComponent as EjsSpeeddial } from "@syncfusion/ej2-vue-
buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
const items = [
  {
      text:'Cut'
  },
  {
      text:'Copy'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
  {
      text:'Paste'
  }
];
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css';
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial'  content='Edit'
target='#targetElement' :items='items'></ejs-speeddial>
    </div>
</template>
<script>
import { SpeedDialComponent } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
    components: {
        'ejs-speeddial': SpeedDialComponent
    },
    data() {
        return {
            items: [
                {
                    text:'Cut'
                },
                {
                    text:'Copy'
                },
                {
                    text:'Paste'
                }
            ]
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

Run the project

To run the project, use the following command:

`bash

Copyright © 2001 -2024 Syncfusion Inc.

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/speed-dial/getting-started-cs1" %}

Getting Started with Syncfusion SpeedDial Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue SpeedDial component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Link to the Video is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

To get started quickly with Vue 3 Speed dial, you can check out this video:

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

Copyright © 2001 -2024 Syncfusion Inc.

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Copyright © 2001 -2024 Syncfusion Inc.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Speed Dial component as an example. To use the Vue Speed Dial component in the project, the @syncfusion/ej2-vue-buttons package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-buttons --save

`

or

`bash

yarn add @syncfusion/ej2-vue-buttons

`

Adding CSS reference for Syncfusion Vue Speed Dial component

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Button component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

Adding Syncfusion Vue Speed Dial component in the application

Follow the below steps to add the Vue Speed Dial component using Composition API or Options API:

1. First, import and register the Speed Dial component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SpeedDialComponent as ejsSpeeddial } from "@syncfusion/ej2-vue-
buttons";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { SpeedDialComponent } from "@syncfusion/ej2-vue-buttons";
//Component registration
export default {
name: "App",
components: {
"ejs-speeddial": SpeedDialComponent
}
}
</script>
```

2. In the template section, define the Speed Dial component with the items property

~/SRC/APP.VUE

```
<template>
<div id="targetElement" style="position:relative;min-height:350px;min-width:
350px;border:1px solid;">
<ejs-speeddial id='speeddial'  content='Edit' target='#targetElement'
:items='items'></ejs-speeddial>
</div>
</template>
```

3. Declare the values for the items property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const items = [
{
text: 'Cut'
},
{
text: 'Copy'
},
{
text: 'Paste'
}];
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data () {
return {
items:[
{
text: 'Cut'
},
{
text: 'Copy'
},
{
text: 'Paste'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}]
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="targetElement" style="position:relative;min-height:350px;min-width:
350px;border:1px solid;">
<ejs-speeddial id='speeddial'  content='Edit' target='#targetElement'
:items='items' @click="click"></ejs-speeddial>
</div>
</template>
<script setup>
import { SpeedDialComponent as ejsSpeeddial } from "@syncfusion/ej2-vue-
buttons";
const items = [
{
text: 'Cut'
},
{
text: 'Copy'
},
{
text: 'Paste'
}];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="targetElement" style="position:relative;min-height:350px;min-
width:350px;border:1px solid;">
<ejs-speeddial id='speeddial'  content='Edit' target='#targetElement'
:items='items'></ejs-speeddial>
</div>
</template>
<script>
import { SpeedDialComponent } from "@syncfusion/ej2-vue-buttons";
export default {
name: "App",
components: {
"ejs-speeddial": SpeedDialComponent
},
data () {
return {
items:[
{
text: 'Cut'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
{
text: 'Copy'
},
{
text: 'Paste'
}]
};
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Copyright © 2001 -2024 Syncfusion Inc.

Items in Vue Speed dial component

The Vue Speed Dial action items can be added by using items property.

| Fields | Type | Description |

|------|------|------------|

| text | string | Defines the text content of SpeedDialItem. |

| iconCss | string | Defines one or more CSS classes to include an icon or image in Speed Dial item. |

| disabled | boolean | Defines whether to enable or disable the SpeedDialItem. |

| id | string | Defines a unique value for the SpeedDialItem which can be used to identify the item in event args. |

| title | string | Defines the title of SpeedDialItem to display tooltip. |

Icons in Speed Dial items

You can customize the icon and text of Speed Dial action items using iconCss and text properties.

Icon only

You can show icon only in SpeedDial items by setting iconCss property. You can show tooltip on hover to show additional details to end-user by setting title property.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial' openIconCss='e-icons e-edit'
closeIconCss='e-icons e-close' target='#targetElement' :items='items'></ejs-
speeddial>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";
    import { enableRipple } from '@syncfusion/ej2-base';
    enableRipple(true);
    Vue.use(SpeedDialPlugin);
    export default {
        data() {
            return {
                items: [
                    { iconCss: 'e-icons e-cut' },
                    { iconCss: 'e-icons e-copy' },
                    { iconCss: 'e-icons e-paste' }
                ]
            };
        }
    }
</script>
<style>
    @import
'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-
base/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/items-cs1" %}

Text only

You can show only text in Speed Dial items by setting text property.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial' content='Edit' target='#targetElement'
:items='items'></ejs-speeddial>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";
    import { enableRipple } from '@syncfusion/ej2-base';
    enableRipple(true);
    Vue.use(SpeedDialPlugin);
    export default {
        data() {
            return {
                items: [
                    { text: 'Cut' },
                    { text: 'Copy' },
                    { text: 'Paste' }
                ]
            };
        }
    }
</script>
<style>
    @import
'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-
base/styles/material.css';
    @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/items-cs2" %}

Icon with text

You can show icon along with text in Speed Dial items by setting iconCss and text properties.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <ejs-speeddial id='speeddial' openIconCss='e-icons e-edit'
closeIconCss='e-icons e-close' content='Edit' target='#targetElement'
:items='items'></ejs-speeddial>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";
    import { enableRipple } from '@syncfusion/ej2-base';
    enableRipple(true);
    Vue.use(SpeedDialPlugin);
    export default {
        data() {
            return {
                items: [
                    { text: 'Cut', iconCss: 'e-icons e-cut' },
                    { text: 'Copy', iconCss: 'e-icons e-copy' },
                    { text: 'Paste', iconCss: 'e-icons e-paste' }
                ]
            };
        }
    }
</script>
<style>
    @import
'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-
base/styles/material.css';
    @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/items-cs3" %}

Disabled

You can disable Speed Dial items by setting disabled property as true.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial' content='Edit' target='#targetElement'
:items='items'></ejs-speeddial>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";
    import { enableRipple } from '@syncfusion/ej2-base';
    enableRipple(true);
    Vue.use(SpeedDialPlugin);
    export default {
        data() {
            return {
                items: [
                    { text: 'Cut', disabled: true },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    { text: 'Copy' },
                    { text: 'Paste' }
                ]
            };
        }
    }
</script>
<style>
    @import
'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-base/styles/material.css';
    @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/items-cs4" %}

Animation

The Speed Dial items can be animated during the opening and closing of the popup action items. You can customize the animation's effect, delay, and duration by setting animation property. By default, Speed Dial animates with a fade effect and supports all speeddialanimation effects.

Below example demonstrates the Speed Dial items with applied Zoom effect.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial' openIconCss='e-icons e-edit' closeIconCss='e-icons e-close' content='Edit' target='#targetElement' :items='items' :animation='animation'></ejs-speeddial>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";
    import { enableRipple } from '@syncfusion/ej2-base';
    enableRipple(true);
    Vue.use(SpeedDialPlugin);
    export default {
        data() {
            return {
                items: [
                    { text: 'Cut', iconCss: 'e-icons e-cut' },
                    { text: 'Copy', iconCss: 'e-icons e-copy' },
                    { text: 'Paste', iconCss: 'e-icons e-paste' }
                ],
                animation: { effect: 'Zoom' }
            };
        }
    }
</script>
<style>
```

```
    @import
'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-
base/styles/material.css';
    @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/items-cs5" %}

Template

The Speed Dial supports to customize the action items and entire pop-up container by setting itemTemplate and popupTemplate properties. For more details about templates, check out the link here.

Positions in Vue Speed dial component

The Speed dial control can be positioned anywhere on the target using the position property. If the target is not defined, then Speed Dial is positioned based on the browser viewport.

The position values of Speed Dial are as follows:

- TopLeft
- TopCenter
- TopRight
- MiddleLeft
- MiddleCenter
- MiddleRight
- BottomLeft
- BottomCenter
- BottomRight

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial' content='Add' position='BottomLeft'
target='#targetElement'></ejs-speeddial>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";
    import { enableRipple } from '@syncfusion/ej2-base';
    enableRipple(true);
    Vue.use(SpeedDialPlugin);
    export default {
        data() {
            return { };
        }
    }
</script>
<style>
    @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
```

```
    @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/position-cs1" %}

Opens items on hover

You can open the Speed Dial action items on mouse hover by setting the opensOnHover property.

`

<template>

<ejs-speeddial id='speeddial' openIconCss='e-icons e-edit' closeIconCss='e-icons e-close' target='#targetElement' opensOnHover= true :items='items'></ejs-speeddial>

</template>

<script>

import Vue from 'vue';

import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";

Vue.use(SpeedDialPlugin);

export default {

data() {

return {

items: [

{ iconCss: 'e-icons e-cut' },

{ iconCss: 'e-icons e-copy' },

{ iconCss: 'e-icons e-paste' }

]

};

}

}

</script>

<style>

@import 'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-base/styles/material.css';

@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';

</style>

`

Programmatically show/hide Speed Dial items

You can open/close the Speed Dial action items programmatically using show and hide methods.

Copyright © 2001 -2024 Syncfusion Inc.

Below example demonstrates open/close action items on button click.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;">
        <ejs-button id="show" class="e-primary" v-
on:click.native="showClick"> Show </ejs-button>
        <ejs-button id="hide" class="e-primary" v-
on:click.native="hideClick"> Hide </ejs-button></div>
        <ejs-speeddial ref="speeddial" id='speeddial' openIconCss='e-icons
e-edit' closeIconCss='e-icons e-close' opensOnHover= true
target='#targetElement' :items='items'></ejs-speeddial>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SpeedDialPlugin, ButtonPlugin } from "@syncfusion/ej2-vue-
buttons";
    import { enableRipple } from '@syncfusion/ej2-base';
    enableRipple(true);
    Vue.use(SpeedDialPlugin);
    Vue.use(ButtonPlugin);
    export default {
        data() {
            return {
                items: [
                    { iconCss: 'e-icons e-cut' },
                    { iconCss: 'e-icons e-copy' },
                    { iconCss: 'e-icons e-paste' }
                ]
            };
        },
        methods: {
            showClick: function () {
                this.$refs.speeddial.show();
            },
            hideClick: function () {
                this.$refs.speeddial.hide();
            }
        }
    }
</script>
<style>
    @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
    @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
    #hide {
        float: right;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/position-cs2" %}

Copyright © 2001 -2024 Syncfusion Inc.

Programmatically refresh the position

You can refresh the position of the Speed Dial using refreshPosition method when the targetposition is changed.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;">
        <ejs-button id="refresh" class="e-primary" v-
on:click.native="refresh"> Refresh </ejs-button></div>
        <ejs-speeddial ref="speeddial" id='speeddial' openIconCss='e-icons
e-edit' closeIconCss='e-icons e-close' target='#targetElement'
position='MiddleRight' :items='items'></ejs-speeddial>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SpeedDialPlugin, ButtonPlugin } from "@syncfusion/ej2-vue-
buttons";
    import { enableRipple } from '@syncfusion/ej2-base';
    enableRipple(true);
    Vue.use(SpeedDialPlugin);
    Vue.use(ButtonPlugin);
    export default {
        data() {
            return {
                items: [
                    { iconCss: 'e-icons e-cut' },
                    { iconCss: 'e-icons e-copy' },
                    { iconCss: 'e-icons e-paste' }
                ]
            };
        },
        methods: {
            refresh: function () {
                document.getElementById("targetElement").style.minHeight =
"300px";
                this.$refs.speeddial.refreshPosition();
            }
        }
    }
</script>
<style>
    @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
    @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
    #hide {
        float: right;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/position-cs3" %}

Copyright © 2001 -2024 Syncfusion Inc.

Display modes in Vue Speed dial component

The action items in Vue Speed Dial can be displayed in Linear and Radial display modes by setting mode property.

Linear display mode

In Linear display mode, Speed Dial action items are displayed in a list-like format either horizontally or vertically. By default, Speed Dial items are displayed in Linear mode.

Direction

You can open the action items on the top, left, up, and down side of the Speed Dial button by setting direction property. The default value is Auto where the action items are displayed based on the positionof the Speed Dial.

The Linear directions of Speed Dial are as follows:

- Left - Action items are displayed on the left side of the button.
- Right - Action items are displayed on the right side of the button.
- Up - Action items are displayed on the top of the button.
- Down - Action items are displayed on the bottom of the button.
- Auto - Action items display direction auto calculated based on position of the Speed Dial. If Speed Dial is position at bottom right, then action items displayed at top.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial' openIconCss='e-icons e-edit'
closeIconCss='e-icons e-close' target='#targetElement' mode='Linear'
direction='Left' :items='items'></ejs-speeddial>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";
    import { enableRipple } from '@syncfusion/ej2-base';
    enableRipple(true);
    Vue.use(SpeedDialPlugin);
    export default {
        data() {
            return {
                items: [
                    { iconCss: 'e-icons e-cut' },
                    { iconCss: 'e-icons e-copy' },
                    { iconCss: 'e-icons e-paste' }
                ]
            };
        }
    }
</script>
<style>
    @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
    @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/radial-cs1" %}

Radial display mode (Radial Menu)

In Radial mode, Speed Dial action items are displayed in a circular pattern like a radial menu. For more details about radial mode, check out the link here.

Radial menu in Vue Speed dial component

The React Speed Dial action items can be displayed in a circular patter like a radial menu by setting mode property. You can customize the direction, startAngle, endAngle and offset by setting radialSettings property.

Radial menu direction

You can open the action items in either clockwise or anticlockwise by setting direction property. The default value is Auto where the action items are displayed based on the position property of the Speed Dial.

`

<template>

<ejs-speeddial id='speeddial' openIconCss='e-icons e-edit' closeIconCss='e-icons e-close' target='#targetElement' mode= 'Radial' :items='items' :radialSettings='radialSettings'></ejs-speeddial>

</template>

<script>

import Vue from 'vue';

import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";

Vue.use(SpeedDialPlugin);

export default {

data() {

return {

items: [

{ iconCss: 'e-icons e-cut' },

{ iconCss: 'e-icons e-copy' },

{ iconCss: 'e-icons e-paste' },

{ iconCss: 'e-icons e-edit' },

{ iconCss: 'e-icons e-save' }

],

radialSettings: { direction: 'AntiClockwise' }

};

}

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
@import 'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
`
```

Radial menu start and end angle

You can modify the start and end angle of action items by setting startAngle and endAngle properties. If the angle is not defined, the action items are displayed based on the position property of the Speed Dial.

```
`
<template>
<ejs-speeddial id='speeddial' openIconCss='e-icons e-edit' closeIconCss='e-icons e-close' target='#targetElement' mode= 'Radial' position='MiddleCenter' :items='items' :radialSettings='radialSettings'></ejs-speeddial>
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(SpeedDialPlugin);
export default {
data() {
return {
items: [
{ iconCss: 'e-icons e-cut' },
{ iconCss: 'e-icons e-copy' },
{ iconCss: 'e-icons e-paste' },
{ iconCss: 'e-icons e-edit' },
{ iconCss: 'e-icons e-save' }
],
radialSettings: { direction: 'AntiClockwise', startAngle: 180, endAngle: 360 }
};
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
@import 'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
`
```

Offset

You can modify the offset distance between action items and Speed Dial button using offset property.

```
`
<template>
<ejs-speeddial id='speeddial' openIconCss='e-icons e-edit' closeIconCss='e-icons e-close' target='#targetElement' mode= 'Radial' :items='items' :radialSettings='radialSettings'></ejs-speeddial>
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(SpeedDialPlugin);
export default {
data() {
return {
items: [
{ iconCss: 'e-icons e-cut' },
{ iconCss: 'e-icons e-copy' },
{ iconCss: 'e-icons e-paste' },
{ iconCss: 'e-icons e-edit' },
{ iconCss: 'e-icons e-save' }
],
radialSettings: { offset: '80px' }
};
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

<style>

@import '../node_modules/@syncfusion/ej2-base/styles/material.css';

@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';

</style>

`

Below example demonstrates the radial menu settings of the Speed Dial.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial' openIconCss='e-icons e-edit'
closeIconCss='e-icons e-close' target='#targetElement' mode= 'Radial'
position='MiddleRight' :items='items'
:radialSettings='radialSettings'></ejs-speeddial>
    </div>
</template>
<script>
    import Vue from 'vue';
    import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";
    import { enableRipple } from '@syncfusion/ej2-base';
    enableRipple(true);
    Vue.use(SpeedDialPlugin);
    export default {
        data() {
            return {
                items: [
                    { iconCss: 'e-icons e-cut' },
                    { iconCss: 'e-icons e-copy' },
                    { iconCss: 'e-icons e-paste' },
                    { iconCss: 'e-icons e-edit' },
                    { iconCss: 'e-icons e-save' }
                ],
                radialSettings: { offset: '80px', direction:
'AntiClockwise', startAngle: 90, endAngle: 270 }
            };
        }
    }
</script>
<style>
    @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
    @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/radial-cs2" %}

Template in Vue Speed dial component

This section explains available templates in SpeedDial component and its usage.

Copyright © 2001 -2024 Syncfusion Inc.

Item template

You can use the itemTemplate property to set a template content for the SpeedDial items. The template content is defined as a child content of itemTemplate tag directive.

APP.VUE

```
<template>
    <div>
      <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
      <ejs-speeddial id='speeddial'  openIconCss='e-icons e-edit'
content="Edit" :itemTemplate ="itemTemplateContent" target='#targetElement'
:items='items'>
      </ejs-speeddial>
    </div>
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SpeedDialPlugin);
export default {
    data() {
        return {
            itemTemplateContent:"<div class='itemlist'><span class='icon
${iconCss}' style='padding:3px'></span><span class='text' style='padding:0
5px'>${text}</span></div>",
            items: [
                { iconCss: 'e-icons e-cut', text: 'Cut' },
                { iconCss: 'e-icons e-copy', text: 'Copy' },
                { iconCss: 'e-icons e-paste', text: 'Paste' }
            ]
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
.itemlist .text {
  padding: 0 5px;
}
.e-speeddial-li .itemlist {
  display: inherit;
  width: 100%;
  border: 1px solid transparent;
  align-items: center;
  padding: 5px;
  border-radius: 500px;
  background-color: rgba(104, 99, 104, 0.1);
  box-shadow: 0 0 4px grey;
}
.e-speeddial-li .itemlist:hover {
  background-color: rgb(224, 224, 224);
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/speed-dial/template-cs1" %}

Popup template

You can use the popupTemplate property to set a template content for popup of SpeedDial component. The template content is defined as a child content of popupTemplate tag directive.

APP.VUE

```
<template>
  <div>
    <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    <ejs-speeddial id='speeddial'  openIconCss='e-icons e-edit'
content="Edit" :popupTemplate="popupTemplateContent"
cssClass='popupSpeedDial' target='#targetElement'>
      <template v-slot:popupTemplateContent>
        <div>
          <div class="speeddial-form">
              <p>Here you can customize your code.</p>
          </div>
        </div>
      </template>
    </ejs-speeddial>
  </div>
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SpeedDialPlugin);
export default {
  data() {
        return {
          popupTemplateContent:"<div><div class='speeddial-form'><p>Here you
can customize your code.</p></div></div>"
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
.speeddial-form {
  width: 200px;
  height: 80px;
  text-align: center;
  border-radius: 15px;
  box-shadow: rgb(0 0 0 / 10%) 0px 10px 15px -3px, rgb(0 0 0 / 5%) 0px 4px
6px -2px;
  background: #f5f5f5;
  padding: 15px;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/speed-dial/template-cs2" %}

Styles in Vue Speed dial component

This section briefs different ways to style SpeedDial component.

SpeedDial button

You can customize the icon and text of Vue Speed Dial button using openIconCss, closeIconCss and content properties.

Icon only

You can use the openIconCss and closeIconCss properties to show icons in speed dial button. You can also show tooltip on hover to show additional details to end-user by setting title attribute.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial' openIconCss='e-icons e-edit'
target='#targetElement'></ejs-speeddial>
    </div>
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SpeedDialPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/styles-cs1" %}

Text only

You can show only text in Speed Dial button by setting content property without setting icon properties..

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial' content='Edit'
target='#targetElement'></ejs-speeddial>
    </div>
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
enableRipple(true);
Vue.use(SpeedDialPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/styles-cs2" %}

Icon with text

You show icon and text in SpeedDial button using openIconCss, closeIconCss and content properties together.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial' content='Edit' openIconCss='e-icons e-
edit' target='#targetElement'></ejs-speeddial>
    </div>
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SpeedDialPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/styles-cs3" %}

Disabled

You can enable or disable the SpeedDial component using disabled property.

`

<template>

<ejs-speeddial id='speeddial' content='Edit' disabled=true></ejs-speeddial>

</template>

<script>

import Vue from 'vue';

import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";

import { enableRipple } from '@syncfusion/ej2-base';

Copyright © 2001 -2024 Syncfusion Inc.

enableRipple(true);

Vue.use(SpeedDialPlugin);

export default {}

</script>

<style>

@import '../node_modules/@syncfusion/ej2-base/styles/material.css';

@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';

</style>

`

cssClass

The Vue Speed Dial supports the following predefined styles that can be defined using the cssClass property. You can customize by setting the cssClass property with the below defined class.

| cssClass | Description |

| -------- | -------- |

| e-primary | Used to represent a primary action. |

| e-outline | Used to represent an appearance of button with outline. |

| e-info | Used to represent an informative action. |

| e-success | Used to represent a positive action. |

| e-warning | Used to represent an action with caution. |

| e-danger | Used to represent a negative action. |

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial' content='Edit' cssClass='e-warning'
target='#targetElement'></ejs-speeddial>
    </div>
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SpeedDialPlugin);
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/speed-dial/styles-cs4" %}

Visible

You can set the Speed Dial button to visible/hidden state using visible property.

`

<template>

<ejs-speeddial id='speeddial' content='Edit' visible=false></ejs-speeddial>

</template>

<script>

import Vue from 'vue';

import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";

import { enableRipple } from '@syncfusion/ej2-base';

enableRipple(true);

Vue.use(SpeedDialPlugin);

export default {}

</script>

<style>

@import '../node_modules/@syncfusion/ej2-base/styles/material.css';

@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';

</style>

`

Tooltip

You can show tooltip on hover to show additional details to end-user by setting title to Speed Dial button.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial'  openIconCss='e-icons e-edit'
target='#targetElement' :items='items'></ejs-speeddial>
    </div>
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SpeedDialPlugin);
export default {
    data() {
        return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            items: [
                { iconCss:'e-icons e-cut', title:'Cut' },
                { iconCss:'e-icons e-copy', title:'Copy' },
                { iconCss:'e-icons e-paste', title:'Paste' }
            ]
        };
    }
}
</script>
<style>
@import
'https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/styles-cs5" %}

Opens on hover

You can use opensOnHover property to open actions items on hover itself. By default action items displayed only when clicking the speed dial button.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial'  openIconCss='e-icons e-edit' closeIconCss='e-icons e-close' opensOnHover=true target='#targetElement' :items='items'></ejs-speeddial>
    </div>
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SpeedDialPlugin);
export default {
    data() {
        return {
            items: [
                { iconCss: 'e-icons e-cut' },
                { iconCss: 'e-icons e-copy' },
                { iconCss: 'e-icons e-paste' }
            ]
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/speed-dial/styles-cs6" %}

Modal in Vue Speed dial component

You can use the modal property to set the Speed Dial as modal which adds an overlay to prevent the background interaction.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial'  openIconCss='e-icons e-edit'
modal=true target='#targetElement' :items='items'></ejs-speeddial>
    </div>
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SpeedDialPlugin);
export default {
    data() {
        return {
            items: [
                { iconCss:'e-icons e-cut'},
                { iconCss:'e-icons e-copy'},
                { iconCss:'e-icons e-paste'}
            ]
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/modal-cs1" %}

Event in Vue Speed dial component

This section explains the Speed Dial events that will be triggered when appropriate actions are performed.

clicked

The SpeedDial component triggers the clicked event with SpeedDialItemEventArgs argument when an action item is clicked. You can use this event to perform the required action.

`

<template>

<ejs-speeddial id='speeddial' content='Edit' target='#targetElement' :items='items'
:clicked="clicked"></ejs-speeddial>

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
Vue.use(SpeedDialPlugin);
export default {
data () {
return {
items: [
{ text: 'Cut' },
{ text: 'Copy' },
{ text: 'Paste' }
]
};
},
methods: {
clicked: function(args) {
//Your required action here
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
`
```

created

The Speed Dial component triggers the created event when SpeedDial component rendering is completed.

```
`
<template>
<ejs-speeddial id='speeddial' content='Edit'  target='#targetElement' :items='items'
:created="created"></ejs-speeddial>
```

```
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
Vue.use(SpeedDialPlugin);
export default {
data () {
return {
items: [
{ text: 'Cut' },
{ text: 'Copy' },
{ text: 'Paste' }
]
};
},
methods: {
created: function(args) {
//Your required action here
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
`
```

beforeOpen

The SpeedDial component triggers the beforeOpen event with SpeedDialBeforeOpenCloseEventArgs argument before the SpeedDial popup is opened.

```
`
<template>
<ejs-speeddial id='speeddial' content='Edit'  target='#targetElement' :items='items'
:beforeOpen="beforeOpen"></ejs-speeddial>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
Vue.use(SpeedDialPlugin);
export default {
data () {
return {
items: [
{ text: 'Cut' },
{ text: 'Copy' },
{ text: 'Paste' }
]
};
},
methods: {
beforeOpen: function(args) {
//Your required action here
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
`
```

onOpen

The SpeedDial component triggers the onOpen event with SpeedDialOpenCloseEventArgs argument when SpeedDial popup is opened.

```
`
<template>
<ejs-speeddial id='speeddial' content='Edit'  target='#targetElement' :items='items'
:onOpen="onOpen"></ejs-speeddial>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
Vue.use(SpeedDialPlugin);
export default {
data () {
return {
items: [
{ text: 'Cut' },
{ text: 'Copy' },
{ text: 'Paste' }
]
};
},
methods: {
onOpen: function(args) {
//Your required action here
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
`
```

beforeClose

The SpeedDial component triggers the beforeClose event with SpeedDialBeforeOpenCloseEventArgs argument before the SpeedDial popup is closed.

```
`
<template>
<ejs-speeddial id='speeddial' content='Edit'  target='#targetElement' :items='items'
:beforeClose="beforeClose"></ejs-speeddial>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
Vue.use(SpeedDialPlugin);
export default {
data () {
return {
items: [
{ text: 'Cut' },
{ text: 'Copy' },
{ text: 'Paste' }
]
};
},
methods: {
beforeClose: function(args) {
//Your required action here
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
`
```

onClose

The SpeedDial component triggers the onClose event with SpeedDialOpenCloseEventArgs argument when SpeedDial popup is closed.

```
`
<template>
<ejs-speeddial id='speeddial' content='Edit'  target='#targetElement' :items='items'
:onClose="onClose"></ejs-speeddial>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
Vue.use(SpeedDialPlugin);
export default {
data () {
return {
items: [
{ text: 'Cut' },
{ text: 'Copy' },
{ text: 'Paste' }
]
};
},
methods: {
onClose: function(args) {
//Your required action here
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
`
```

beforeItemRender

The SpeedDial component triggers the beforeItemRender event with SpeedDialItemEventArgs argument for each SpeedDialItem once it is rendered.

```
`
<template>
<ejs-speeddial id='speeddial' content='Edit'  target='#targetElement' :items='items'
:beforeItemRender="beforeItemRender"></ejs-speeddial>
```

Copyright © 2001 -2024 Syncfusion Inc.

</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(SpeedDialPlugin);
export default {
data () {
return {
items: [
{ text: 'Cut' },
{ text: 'Copy' },
{ text: 'Paste' }
]
};
},
methods: {
beforeItemRender: function(args) {
//Your required action here
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
`

Below example demonstrates the clicked event of the Speed Dial component.

APP.VUE

```
<template>
    <div>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <ejs-speeddial id='speeddial' content='Edit'
target='#targetElement' :items='items' :clicked="clicked"></ejs-speeddial>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { SpeedDialPlugin  } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SpeedDialPlugin);
export default {
    data () {
        return {
            items: [
                { text: 'Cut' },
                { text: 'Copy' },
                { text: 'Paste' }
            ]
        };
    },
    methods: {
      clicked: function(args) {
          alert(args.item.text+" is clicked");
      }
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/speed-dial/action-cs1" %}

Accessibility in Vue Speed dial component

Accessibility is achieved in the Speed Dial component through WAI-ARIA standard and keyboard navigations. The Speed Dial component can be effectively accessed through assistive technologies such as screen readers.

Keyboard interaction

The Speed Dial component is interactive with below keyboard shortcuts.

| Keyboard shortcuts | Actions |

|----------|----------|

| Enter | Open/close the menu. If a SpeedDial item is focused, should triggers the clicked event for the item. |

| Up-arrow | Navigates up or to the previous menu item. |

| Left-Arrow | Navigates left or to the previous menu item. |

| Down-Arrow | Navigates down or to the previous menu item. |

| Right-Arrow | Navigates right or to the previous menu item. |

| Home | Navigates to the first menu item. |

| End | Navigates to the last menu item. |

Copyright © 2001 -2024 Syncfusion Inc.

| Esc | Closes the menu. |

ARIA attributes

The following ARIA attributes are used in SpeedDial component based on its state.

| Properties | Functionality |

| ---------- | ---------- |

| role | This attribute is added to the input element to describe the actual role. |

| aria-label | Attribute provides the text label with some default description for the SpeedDial and its items. |

| aria-expanded | It indicates whether the SpeedDial current state is expanded or collapsed. |

| aria-haspopup | It indicates whether the SpeedDial has popup items or not. |

| aria-controls | Attribute is set to the SpeedDial button and it points to the corresponding content. |

| aria-disabled | It indicates the disabled state of the SpeedDial and its items. |

Spinner

Getting Started with the Vue Spinner Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Spinner component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

vue create quickstart

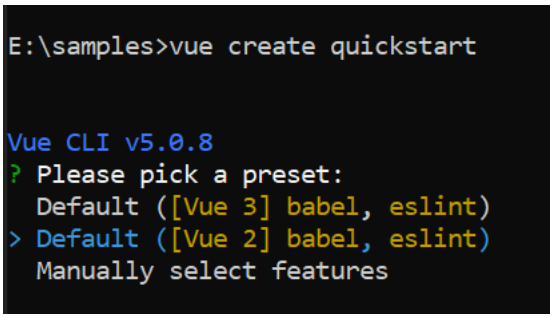
cd quickstart

yarn run serve

`

Copyright © 2001 -2024 Syncfusion Inc.

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

Install the @syncfusion/ej2-vue-popups package by running the following command:

`bash

npm install @syncfusion/ej2-vue-popups --save

`

or

`bash

yarn add @syncfusion/ej2-vue-popups

`

Add Syncfusion Vue component

Follow the below steps to add the Vue Spinner component using Composition API or Options API:

1\. First, import and register the Spinner component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { onMounted } from 'vue';
import { createSpinner, showSpinner, hideSpinner } from '@syncfusion/ej2-
vue-popups';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { createSpinner, showSpinner, hideSpinner } from '@syncfusion/ej2-
vue-popups';
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
   <div id="app">
    </div>
</template>
<script setup>
import { onMounted } from 'vue';
import { createSpinner, showSpinner, hideSpinner } from '@syncfusion/ej2-
vue-popups';
onMounted (() => {
    //createSpinner() method is used to create spinner
    createSpinner({
    // Specify the target for the spinner to show
    target: document.getElementById('app')
  });

  // showSpinner() will make the spinner visible
  showSpinner(document.getElementById('app'));
  setInterval(() => {
    // hideSpinner() method used hide spinner
    hideSpinner(document.getElementById('app'));
  }, 100000);
});
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
   <div id="app">
    </div>
</template>
<script>
import { createSpinner, showSpinner, hideSpinner } from '@syncfusion/ej2-
vue-popups';
export default {
  name: 'app',
  mounted: function() {
    //createSpinner() method is used to create spinner
    createSpinner({
      // Specify the target for the spinner to show
      target: document.getElementById('app')
    });
    // showSpinner() will make the spinner visible
    showSpinner(document.getElementById('app'));
    setInterval(function(){
      // hideSpinner() method used hide spinner
      hideSpinner(document.getElementById('app'));
    }, 100000);
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
 @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/spinner/getting-started-cs1" %}

Getting Started with the Vue Spinner Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Spinner component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

Copyright © 2001 -2024 Syncfusion Inc.

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

Copyright © 2001 -2024 Syncfusion Inc.

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

To use the Vue Spinner component in the project, the @syncfusion/ej2-vue-popups package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-popups --save

`

or

`bash

yarn add @syncfusion/ej2-vue-popups

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Spinner component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Adding Syncfusion Vue3 Spinner component in the application

You have completed all the necessary configurations needed for rendering the Syncfusion Vue component. Now, you are going to add the Spinner.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app"></div>
</template>
<script>
import { onMounted } from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { createSpinner, showSpinner, hideSpinner } from "@syncfusion/ej2-vue-popups";
export default {
name: "App",
setup() {
onMounted(() => {
const appElement = document.getElementById("app");
const spinner = createSpinner({
target: appElement,
});
showSpinner(appElement);
setInterval(() => {
hideSpinner(appElement);
}, 100000);
});
return {};
},
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app"></div>
</template>
<script>
import { createSpinner, showSpinner, hideSpinner } from "@syncfusion/ej2-vue-popups";
export default {
name: "App",
mounted: function () {
//createSpinner() method is used to create spinner
createSpinner({
// Specify the target for the spinner to show
target: document.getElementById("app"),
});
// showSpinner() will make the spinner visible
showSpinner(document.getElementById("app"));
setInterval(function () {
// hideSpinner() method used hide spinner
hideSpinner(document.getElementById("app"));
}, 100000);
},
data: function () {
return {};
},
methods: {},
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

Template in Vue Spinner component

You can use custom templates on the Spinner instead of the default Spinner by specifying the template in the setSpinner method.

The following steps explains you on how to define template for Spinner.

- Import the setSpinner method from ej2-vue-popups library into your vue as shown in below.

`ts

import { setSpinner } from '@syncfusion/ej2-vue-popups';

`

- Pass your custom template to the setSpinner method like as below.

`ts

// Specify the template content to be displayed in the Spinner

setSpinner({ template: ' '});

`

You should set the template to the Spinner before creating the respective Essential JS 2 component.

Also,until we replace setSpinner template, the further Essential JS 2 component rendering is created with given template only.

Copyright © 2001 -2024 Syncfusion Inc.

- Now, render the Essential JS 2 component. It's render the Spinner with the template specified in the setSpinner method.

In the below sample, we have rendered the Grid component with custom Spinner using setSpinner method.

You have to define the styles for the template in index.css.

APP.VUE

```
<template>
   <div id="app">
     <ejs-grid  ref='elementRef1' id='grid1'  :dataSource="data"
:created='onGridCreated'>
     <e-columns>
      <e-column field='OrderID' headerText='Order ID' textAlign='Right'
width=90></e-column>
      <e-column field='CustomerID' headerText='Customer ID' width=120></e-
column>
      <e-column field='Freight' headerText='Freight' textAlign='Right'
format='C2' width=90></e-column>
     </e-columns>
    </ejs-grid>
        <br/>
       <br/>
      <ejs-grid  ref='elementRef2' id='grid2'  :dataSource="data" >
      <e-columns>
      <e-column field='OrderID' headerText='Order ID' textAlign='Right'
width=90></e-column>
      <e-column field='CustomerID' headerText='Customer ID' width=120></e-
column>
      <e-column field='Freight' headerText='Freight' textAlign='Right'
format='C2' width=90></e-column>
     </e-columns>
     </ejs-grid>
    </div>
</template>
<script>
import Vue from "vue";
import { GridPlugin} from "@syncfusion/ej2-vue-grids";
import { setSpinner } from '@syncfusion/ej2-vue-popups';
import { orderDetails } from './datasource.ts';
Vue.use(GridPlugin);
export default {
  name: 'app',
  data: function(){
        return {
           data: orderDetails.slice(0, 7)
        }
   },
  methods: {
       onGridCreated: function(args){
        setSpinner({ template: '<div style="width:100%;height:100%"
class="custom-rolling"><div></div></div>' });
       },
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
          .e-spinner-pane.e-spin-hide {
              display: block;
          }
          @keyframes custom-rolling {
              0% {
                  -webkit-transform: translate(-50%, -50%) rotate(0deg);
                  transform: translate(-50%, -50%) rotate(0deg);
              }
              100% {
                  -webkit-transform: translate(-50%, -50%) rotate(360deg);
                  transform: translate(-50%, -50%) rotate(360deg);
              }
            }
            @-webkit-keyframes custom-rolling {
              0% {
                  -webkit-transform: translate(-50%, -50%) rotate(0deg);
                  transform: translate(-50%, -50%) rotate(0deg);
              }
              100% {
                  -webkit-transform: translate(-50%, -50%) rotate(360deg);
                  transform: translate(-50%, -50%) rotate(360deg);
              }
            }
            .custom-rolling {
              position: relative;
            }
            .custom-rolling div,
            .custom-rolling div:after {
              border: 16px solid #51CACC;
              border-radius: 50%;
              border-top-color: transparent;
              height: 160px;
              position: absolute;
              width: 160px;
            }
            .custom-rolling div {
              -webkit-animation: custom-rolling 1.3s linear infinite;
              animation: custom-rolling 1.3s linear infinite;
              top: 100px;
              left: 100px;
            }
            .custom-rolling div:after {
              -ms-transform: rotate(90deg);
              -webkit-transform: rotate(90deg);
              transform: rotate(90deg);
            }
            .custom-rolling {
              -webkit-transform: translate(-31px, -31px) scale(0.31)
translate(31px, 31px);
              height: 62px !important;
              transform: translate(-31px, -31px) scale(0.31) translate(31px,
31px);
              width: 62px !important;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            }
</style>
```

{% previewsample "page.domainurl/code-snippet/spinner/setspinner-cs1" %}

Types in Vue Spinner component

By default, the Spinner is loaded in the applicable Essential JS 2 component based on the theme imported into the page. Based on the theme, the type is set to the Spinner.

The available types are:

- Material
- Fabric
- Bootstrap

You can change the Essential JS 2 component spinner type by passing the type of the spinner as parameter to the setSpinner method like as below.

`ts

// Specify the type of the Spinner to be displayed

setSpinner({ type: 'Bootstrap'});

`

After Essential JS 2 component creation only, you can change the Essential JS 2 component spinner type.

APP.VUE

```
<template>
   <div id="app">
     <ejs-grid  ref='elementRef' id='grid'  :dataSource="data"
:created='onGridCreated'>
     <e-columns>
      <e-column field='OrderID' headerText='Order ID' textAlign='Right'
width=90></e-column>
      <e-column field='CustomerID' headerText='Customer ID' width=120></e-
column>
      <e-column field='Freight' headerText='Freight' textAlign='Right'
format='C2' width=90></e-column>
     </e-columns>
    </ejs-grid>
    </div>
</template>
<script>
import Vue from "vue";
import { GridPlugin} from "@syncfusion/ej2-vue-grids";
import { setSpinner } from '@syncfusion/ej2-vue-popups';
import { orderDetails } from './datasource.ts';
Vue.use(GridPlugin);
export default {
  name: 'app',
  data: function(){
        return {
           data: orderDetails.slice(0, 7)
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        }
    },
  methods: {
        onGridCreated: function(args){
            setSpinner({ type: 'Bootstrap' });
        },
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
.e-spinner-pane.e-spin-hide {
        display: block;
      }
</style>
```

{% previewsample "page.domainurl/code-snippet/spinner/setspinner-cs2" %}

Style in Vue Spinner component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the spinner

Use the following CSS to customize the spinner stroke color.

Material theme

`css

.e-spinner-pane .e-spinner-inner .e-spin-material {

stroke: green;

}

`

Fabric theme

`css

.e-spinner-pane .e-spinner-inner .e-spin-fabric {

stroke: green;

}

`

Bootstrap theme

`css

.e-spinner-pane .e-spinner-inner .e-spin-bootstrap {

fill: green;

stroke: green;

}

`

Copyright © 2001 -2024 Syncfusion Inc.

Bootstrap4 theme

```css
.e-spinner-pane .e-spinner-inner .e-spin-bootstrap4 {
stroke: green;
}
```

High Contrast theme

```css
.e-spinner-pane .e-spinner-inner .e-spin-high-contrast .e-path-arc {
stroke: green;
}
```

SplitButton

Getting Started with the Vue Split button Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Split button component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The list of dependencies required to use the SplitButton component in your application is given as follows:

```js
|-- @syncfusion/ej2-vue-splitbuttons
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-splitbuttons
|-- @syncfusion/ej2-vue-base
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
```

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

```bash
npm install -g @vue/cli
vue create quickstart
```

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

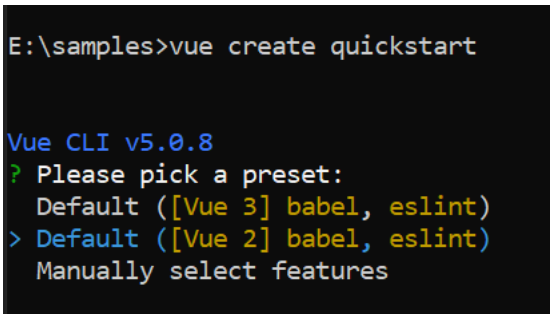
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Split button component as an example. Install the @syncfusion/ej2-vue-splitbuttons package by running the following command:

`bash

npm install @syncfusion/ej2-vue-splitbuttons --save

`

or

`bash

yarn add @syncfusion/ej2-vue-splitbuttons

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

Copyright © 2001 -2024 Syncfusion Inc.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Split button component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Split button component using Composition API or Options API:

1\. First, import and register the Split button component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SplitButtonComponent as EjsSplitbutton } from "@syncfusion/ej2-vue-splitbuttons";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SplitButtonComponent } from "@syncfusion/ej2-vue-splitbuttons";
export default {
components: {
'ejs-splitbutton': SplitButtonComponent
}
}
</script>
```

2\. In the template section, define the Split button component with the items and content property.

~/SRC/APP.VUE

```
<template>
<ejs-splitbutton :items='items' content='Paste'></ejs-splitbutton>
</template>
```

3\. Declare the value for the items property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script>
const items = [
{
text: 'Cut'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
{
text: 'Copy'
},
{
text: 'Paste'
}];
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data () {
return {
items:[
{
text: 'Cut'
},
{
text: 'Copy'
},
{
text: 'Paste'
}]
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-splitbutton :items='items' content='Paste'></ejs-splitbutton>
</template>
<script setup>
import { SplitButtonComponent as EjsSplitbutton } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
const items = [
{
    text: 'Cut'
},
{
    text: 'Copy'
},
{
    text: 'Paste'
}];
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-splitbutton :items='items' content='Paste'></ejs-splitbutton>
</template>
<script>
import { SplitButtonComponent } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
    components: {
        'ejs-splitbutton': SplitButtonComponent
    },
    data () {
        return {
            items:[
            {
                text: 'Cut'
            },
            {
                text: 'Copy'
            },
            {
                text: 'Paste'
            }]
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/split-button/default-cs2" %}

See Also

- SplitButton with icons

Getting Started with the Vue SplitButton Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue SplitButton component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

Copyright © 2001 -2024 Syncfusion Inc.

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue SplitButton component as an example. To use the Vue SplitButton component in the project, the @syncfusion/ej2-vue-splitbuttons package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-splitbuttons --save

Copyright © 2001 -2024 Syncfusion Inc.

`

or

`bash

yarn add @syncfusion/ej2-vue-splitbuttons

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the SplitButton component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue SplitButton component using Composition API or Options API:

1.First, import and register the SplitButton component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SplitButtonComponent as EjsSplitbutton } from "@syncfusion/ej2-vue-splitbuttons";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SplitButtonComponent } from "@syncfusion/ej2-vue-splitbuttons";
//Component registration
export default {
name: "App",
components: {
"ejs-splitbutton": SplitButtonComponent
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

2.In the template section, define the SplitButton component with the items property and content property.

~/SRC/APP.VUE

```
<template>
<ejs-splitbutton :items='items' content='content'></ejs-splitbutton>
</template>
```

3.Declare the values for the items and content property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const content = 'Paste';
const items:[
{
text: 'Cut'
},
{
text: 'Copy'
},
{
text: 'Paste'
}
];
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
content = 'Paste',
items:[
{
text: 'Cut'
},
{
text: 'Copy'
},
{
text: 'Paste'
}
],
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<ejs-splitbutton :items='items' content='content'></ejs-splitbutton>
</template>
<script setup>
import { SplitButtonComponent as EjsSplitbutton } from "@syncfusion/ej2-vue-splitbuttons";
const content = 'Paste';
const items:[
{
text: 'Cut'
},
{
text: 'Copy'
},
{
text: 'Paste'
}
];
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-splitbutton :items='items' content='content'></ejs-splitbutton>
</template>
<script>
import { SplitButtonComponent } from "@syncfusion/ej2-vue-splitbuttons";
// Component registration
export default {
name: "App",
// Declaring component and its directives
components: {
"ejs-splitbutton": SplitButtonComponent
},
// Bound properties declarations
data() {
return {
content = 'Paste',
items:[
{
text: 'Cut'
},
{
text: 'Copy'
},
{
text: 'Paste'
}
],
};
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

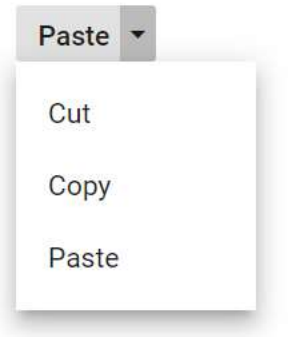
or

`bash

yarn run dev

`

The output will appear as follows:



Sample: vue-3-split-button-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Icons and separator in Vue Split button component

SplitButton icons

SplitButton can have an icon to provide the visual representation of the action. To place the icon on a SplitButton, set the iconCss property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the SplitButton. You can customize the icon's position by using the iconPosition property.

Copyright © 2001 -2024 Syncfusion Inc.

In the following example, the SplitButton with default iconPosition and iconPosition as Top is showcased:

APP.VUE

```
<template>
    <div>
        <ejs-splitbutton :items='items' iconCss='e-sb-icons e-
paste'>Paste</ejs-splitbutton>
        <ejs-splitbutton :items='items' iconCss='e-sb-icons e-paste'
iconPosition="Top">Paste</ejs-splitbutton>
    </div>
</template>
<script>
import Vue from 'vue';
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SplitButtonPlugin);
export default {
    data () {
        return {
            items:[
            {
                text: 'Undo'
            },
            {
                text: 'Redo'
            },
            {
                text: 'Cut'
            },
            {
                text: 'Copy'
            },
            {
                text: 'Paste'
            }]
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@font-face {
    font-family: 'ddb-icons';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj0gSRkAAAEoAAAAVmNtYXDnE+dkAAABlAAAADxnbHlmlh3
3NQAAAdwAAAJMaGVhZBKOK9sAAADQAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAYAAAAA
UbG9jYQGOAegAAAHQAAAADG1heHABEwBlAAABCAAAACBuYW1l1LBM9QAABCgAAAI9cG9zdMJntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADygABAAAAAAAAAAAAAAAAAAAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDygPsAAAACAACAAAAAAAAAAEAAAAFAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAABAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAAEAAgADAAQAAAAAAI4AwgEAASYAAwAA//oDNQPsAA4AHQBYAAAlHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMhYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFxo6IBkNCQIHCy8bCQG9BwIJDRkgOhoXHwoKGi/+TR1RDyE
OIxo+ExckFAQMFikwVhcMBwYlFRYkBwcMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAycCf2TPI0w2HBUmDg0sOzsaKQ4ONzcniyYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEAAAAAAOqA+kABQANABcAHwAAARUzFSERAyERIzU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVL+PHBHCBs
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAAPKA+oAAgATAAABFxEBDgEHHgEXETMRMxE
zETM1IQL+zP1abpADA5t0f2F+XP41AfbMAZgBJwmYcHSbA/48A2r81gNqfgAAAAASAN4AAQAAAAA
AAAABAAAAAQAAAAAAAQAJAAEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAAACgAsADcAAQAAAAAACwASAGMAAwABBAkAAAACAHU
AAwABBAkAAQASAHcAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAAwABBAkACwAkATsgZGRiLWljb25zUmVndWxhcmR
kYi1pY29uc2RkYi1pY29uc1ZlcnNpb24gMS4wZGRiLWljb25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgBzAFY
AZQByAHMAaQBvAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwBlAG4AZQB
yAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
OcGFyYS1tYXJrLS0tMDMAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-sb-icons {
    font-family: 'ddb-icons' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-paste::before {
  content: '\e701';
}
</style>
```

{% previewsample "page.domainurl/code-snippet/split-button/default-cs8" %}

The Essential JS 2 provides a set of icons that can be loaded by applying e-icons class name to the element. You can also use third party icons on the SplitButton using the iconCssproperty.

Vertical button

Vertical Button in SplitButton can be achieved by adding e-vertical class using cssClass property.

The following example illustrates how to provide vertical support in SplitButton component.

APP.VUE

```
<template>
    <ejs-splitbutton :items='items' iconCss='e-sb-icons e-paste'
cssClass='e-vertical' iconPosition="Top">Paste</ejs-splitbutton>
</template>
<script>
import Vue from 'vue';
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SplitButtonPlugin);
export default {
    data () {
        return {
            items:[
            {
                text: 'Undo'
            },
            {
                text: 'Redo'
            },
            {
                text: 'Cut'
            },
            {
                text: 'Copy'
            },
            {
                text: 'Paste'
            }]
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
  .e-split-btn-wrapper{
   margin: 20px 20px 5px 5px;
  }
@font-face {
    font-family: 'ddb-icons';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj0gSRkAAAEoAAAAVmNtYXDnE+dkAAABlAAAADxnbHlmlh3
3NQAAAdwAAAJMaGVhZBKOK9sAAADQAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAYAAAAA
UbG9jYQGOAegAAAHQAAAADG1heHABEwBlAAABCAAAACBuYW1l1LBM9QAABCgAAAI9cG9zdMJntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADygABAAAAAAAAAAAAAAAAAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDygPsAAAACAACAAAAAAAAAAEAAAAFAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAAEAAgADAAQAAAAAAI4AwgEAASYAAwAA//oDNQPsAA4AHQBYAAAlHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMhYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFxo6IBkNCQIHCy8bCQG9BwIJDRkgOhoXHwoKGi/+TR1RDyE
OIxo+ExckFAQMFikwVhcMBwYlFRYkBwcMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
```

```
JCBsSKxYhJ0gWKxIaCQknUEILAycCf2TPI0w2HBUmDg0sOzsaKQ4ONzcniyYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEAAAAAAOqA+kABQANABcAHwAAARUzFSERAyERIzU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVL+PHBHCBs
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAAPKA+oAAgATAAABFxEBDgEHHgEXETMRMxE
zETM1IQL+zP1abpADA5t0f2F+XP41AfbMAZgBJwmYcHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAABAAAAAQAAAAAAAQAJAAEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAAACgAsADcAAQAAAAAACwASAGMAAwABBAkAAAACAHU
AAwABBAkAAQASAHcAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAAwABBAkACwAkATsgZGRiLWljb25zUmVndWxhcmR
kYi1pY29uc2RkYi1pY29uc1ZlcnNpb24gMS4wZGRiLWljb25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgBzAFY
AZQByAHMAaQBvAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwBlAG4AZQB
yAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxBGZvbnQOcGFyYS1tYXJrLS0tMDMAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-sb-icons {
    font-family: 'ddb-icons' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-paste::before {
  content: '\e701';
}
</style>
```

{% previewsample "page.domainurl/code-snippet/split-button/default-cs9" %}

The Essential JS 2 provides a set of icons that can be loaded by applying e-icons class name to the element. You can also use third party icons on the SplitButton using the iconCss property.

Separator

The Separators are the horizontal lines that are used to separate the popup items. You cannot select the separators. You can enable separators to group the popup items using the separator property.

The following example illustrates how to enable separator support in SplitButton component.

APP.VUE

```
<template>
    <ejs-splitbutton :items='items' iconCss='e-sb-icons e-paste'>Paste</ejs-
splitbutton>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SplitButtonPlugin);
export default {
    data () {
        return {
            items:[
            {
                text: 'Cut',
                iconCss: 'e-sb-icons e-cut'
            },
            {
                text: 'Copy',
                iconCss: 'e-icons e-copy'
            },
            {
                text: 'Paste',
                iconCss: 'e-sb-icons e-paste'
            },
            {
                separator: true
            },
            {
                text: 'Font',
                iconCss: 'e-sb-icons e-font'
            },
            {
                text: 'Paragraph',
                iconCss: 'e-sb-icons e-paragraph'
            }]
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
  .e-split-btn-wrapper{
   margin: 20px 20px 5px 5px;
  }
@font-face {
    font-family: 'ddb-icons';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj0gSRkAAAEoAAAAVmNtYXDnE+dkAAABlAAAADxnbHlmlh3
3NQAAAdwAAAJMaGVhZBKOK9sAAADQAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAYAAAAA
UbG9jYQGOAegAAAHQAAAADG1heHABEwBlAAABCAAAACBuYW1l1LBM9QAABCgAAAI9cG9zdMJntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADygABAAAAAAAAAAAAAAAAAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDygPsAAAACAACAAAAAAAAAAEAAAAFAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
```

```
AAAEABAAAAAEAAgADAAQAAAAAAI4AwgEAASYAAwAA//oDNQPsAA4AHQBYAAAlHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMhYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFxo6IBkNCQIHCy8bCQG9BwIJDRkgOhoXHwoKGi/+TR1RDyE
OIxo+ExckFAQMFikwVhcMBwYlFRYkBwcMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAycCf2TPI0w2HBUmDg0sOzsaKQ4ONzcniyYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEAAAAAAOqA+kABQANABcAHwAAARUzFSERAyERIzU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVL+PHBHCBs
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAAPKA+oAAgATAAABFxEBDgEHHgEXETMRMxE
zETM1IQL+zP1abpADA5t0f2F+XP41AfbMAZgBJwmYcHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAABAAAAAQAAAAAAAQAJAAEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAAACgAsADcAAQAAAAAACwASAGMAAwABBAkAAAACAHU
AAwABBAkAAQASAHcAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAAwABBAkACwAkATsgZGRiLWljb25zUmVndWxhcmR
kYi1pY29uc2RkYi1pY29uc1ZlcnNpb24gMS4wZGRiLWljb25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgBzAFY
AZQByAHMAaQBvAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwBlAG4AZQB
yAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxBGZvbnQ
OcGFyYS1tYXJrLS0tMDMAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-sb-icons {
    font-family: 'ddb-icons' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
    .e-cut::before {
        content: '\e700';
    }
    .e-copy::before {
        content: '\e70a';
    }
    .e-paste::before {
        content: '\e701';
    }
    .e-font::before {
        content: '\e702';
    }
    .e-paragraph::before {
        content: '\e703';
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/split-button/default-cs10" %}

Copyright © 2001 -2024 Syncfusion Inc.

See Also

- SplitButton popup with icons

Popup items in Vue Split button component

Icons

The Popup action item have an icon or image to provide visual representation of the action. To place the icon on a popup item, set the iconCss property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the popup action item.

In the following sample, the icons for Cut, Copy, and Paste menu items are added using the iconCss property.

APP.VUE

```
<template>
    <ejs-splitbutton :items='items' iconCss='e-sb-icons e-paste'>Paste</ejs-
splitbutton>
</template>
<script>
import Vue from 'vue';
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SplitButtonPlugin);
export default {
    data () {
        return {
            items:[
            {
                text: 'Cut',
                iconCss: 'e-sb-icons e-cut'
            },
            {
                text: 'Copy',
                iconCss: 'e-icons e-copy'
            },
            {
                text: 'Paste',
                iconCss: 'e-sb-icons e-paste'
            }]
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
  .e-split-btn-wrapper{
   margin: 20px 20px 5px 5px;
  }
@font-face {
    font-family: 'ddb-icons';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj0gSRkAAAEoAAAAVmNtYXDnE+dkAAABlAAAADxnbHlmlh3
3NQAAAdwAAAJMaGVhZBKOK9sAAADQAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAYAAAAA
UbG9jYQGOAegAAAHQAAAADG1heHABEwBlAAABCAAAACBuYW1l1LBM9QAABCgAAAI9cG9zdMJntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADygABAAAAAAAAAAAAAAAAAAAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDygPsAAAACAACAAAAAAAAAAEAAAAFAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAAEAAgADAAQAAAAAAI4AwgEAASYAAwAA//oDNQPsAA4AHQBYAAAlHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMhYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFxo6IBkNCQIHCy8bCQG9BwIJDRkgOhoXHwoKGi/+TR1RDyE
OIxo+ExckFAQMFikwVhcMBwYlFRYkBwcMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAycCf2TPI0w2HBUmDg0sOzsaKQ4ONzcniyYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEAAAAAAOqA+kABQANABcAHwAAARUzFSERAyERIzU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVL+PHBHCBs
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAAPKA+oAAgATAAABFxEBDgEHHgEXETMRMxE
zETM1IQL+zP1abpADA5t0f2F+XP41AfbMAZgBJwmYcHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAABAAAAAQAAAAAAAQAJAAEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAAACgAsADcAAQAAAAAACwASAGMAAwABBAkAAAACAHU
AAwABBAkAAQASAHcAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAAwABBAkACwAkATsgZGRiLWljb25zUmVndWxhcmR
kYi1pY29uc2RkYi1pY29uc1ZlcnNpb24gMS4wZGRiLWljb25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgBzAFY
AZQByAHMAaQBvAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwBlAG4AZQB
yAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
OcGFyYS1tYXJrLS0tMDMAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-sb-icons {
    font-family: 'ddb-icons' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-paste::before {
  content: '\e701';
}
.e-cut::before {
  content: '\e700';
}
.e-copy::before {
  content: '\e70a';
}
```

```
</style>
```

{% previewsample "page.domainurl/code-snippet/split-button/default-cs11" %}

Template

Item templating

Popup items can be customized by using the beforeItemRender event. The item render event triggers while rendering each Popup action item. The event argument will be used to identify the action item and customize it based on the requirement.

In the following example, the icons in each li items is right aligned by appending span element while li rendering:

APP.VUE

```
<template>
    <ejs-splitbutton :items='items' iconCss='e-sb-icons e-paste'
:beforeItemRender='onBeforeItemRender'></ejs-splitbutton>
</template>
<script>
import Vue from 'vue';
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple, createElement } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SplitButtonPlugin);
export default {
    data () {
        return {
            items:[
            {
                text: 'Cut',
            },
            {
                text: 'Copy',
            },
            {
                text: 'Paste',
            }]
        };
    },
    methods: {
        onBeforeItemRender: function(args){
            var shortCutSpan = createElement('span');
            var text = args.item.text;
            args.element.appendChild(shortCutSpan);
            shortCutSpan.setAttribute('class','shortcut');
            var clsName = (text == 'Copy') ? 'e-icons' : 'e-sb-icons';
            shortCutSpan.classList.add(clsName);
            (text === 'Cut') ? shortCutSpan.classList.add('e-cut') : (text
=== 'Paste') ? shortCutSpan.classList.add('e-paste') :
shortCutSpan.classList.add('e-copy');
        }
    }
}
</script>
<style>
```

```
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
  .e-split-btn-wrapper{
   margin: 20px 20px 5px 5px;
  }
@font-face {
    font-family: 'ddb-icons';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj0gSRkAAAEoAAAAVmNtYXDnE+dkAAABlAAAADxnbHlmlh3
3NQAAAdwAAAJMaGVhZBKOK9sAAADQAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAYAAAAA
UbG9jYQGOAegAAAHQAAAADG1heHABEwBlAAABCAAAACBuYW1l1LBM9QAABCgAAAI9cG9zdMJntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADygABAAAAAAAAAAAAAAAAAAAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDygPsAAAACAACAAAAAAAAAAEAAAAFAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAAAEAAgADAAQAAAAAAI4AwgEAASYAAwAA//oDNQPsAA4AHQBYAAAlHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMhYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFxo6IBkNCQIHCy8bCQG9BwIJDRkgOhoXHwoKGi/+TR1RDyE
OIxo+ExckFAQMFikwVhcMBwYlFRYkBwcMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAycCf2TPI0w2HBUmDg0sOzsaKQ4ONzcniyYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEAAAAAAOqA+kABQANABcAHwAAARUzFSERAyERIzU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVL+PHBHCBs
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAAPKA+oAAgATAAABFxEBDgEHHgEXETMRMxE
zETM1IQL+zP1abpADA5t0f2F+XP41AfbMAZgBJwmYcHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAABAAAAAQAAAAAAAQAJAAEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAAACgAsADcAAQAAAAAACwASAGMAAwABBAkAAAACAHU
AAwABBAkAAQASAHcAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAAwABBAkACwAkATsgZGRiLWljb25zUmVndWxhcmR
kYi1pY29uc2RkYi1pY29uc1ZlcnNpb24gMS4wZGRiLWljb25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgBzAFY
AZQByAHMAaQBvAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwBlAG4AZQB
yAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
OcGFyYS1tYXJrLS0tMDMAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-sb-icons {
    font-family: 'ddb-icons' !important;
    speak: none;
    font-size: 14px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
```

```
.e-paste::before {
  content: '\e701';
}
.e-cut::before {
  content: '\e700';
}
.e-copy::before {
  content: '\e70a';
}
</style>
```

{% previewsample "page.domainurl/code-snippet/split-button/default-cs12" %}

Popup templating

The whole popup can be customized as per the requirement and it can be customized by handling it in target property.

In the following sample, the whole popup item is customized as color palette by giving div as target and it can be achieved using target property.

APP.VUE

```
<template>
    <div>
        <div id='dropdowntarget'>
            <div id= "first">
                <div id='black'></div>
                <div id='red'></div>
                <div id='green'></div>
                <div id='gray'></div>
                <div id='blue'></div>
                <div id='violet'></div>
                <div id='brown'></div>
                <div id='darkgoldenrod'></div>
                <div id='aquamarine'></div>
            </div>
        </div>
        <ejs-splitbutton  target='#dropdowntarget' iconCss='e-sb-icons e-
color'></ejs-splitbutton>
    </div>
</template>
<script>
import Vue from 'vue';
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SplitButtonPlugin);
export default {
    data () {
        return {
            items:[
            {
                text: 'Cut',
                iconCss: 'e-icons e-cut'
            },
            {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                text: 'Copy',
                iconCss: 'e-icons e-copy'
            },
            {
                text: 'Paste',
                iconCss: 'e-icons e-paste'
            }]
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
    .e-split-btn-wrapper{
        margin: 20px 20px 5px 5px;
    }
    @font-face {
        font-family: 'paint';
        src:
        url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj0gSRIAAAEoAAAAVmNtYXDnEOdVAAABiAAAADZnbHlmIZD
+uwAAAcgAAADMaGVhZBKhhHQAAADQAAAANmhoZWEHjANrAAAArAAAACRobXR4B+j/8wAAAYAAAAA
IbG9jYQBmAAAAAAHAAAAABm1heHABDgBKAAABCAAAACBuYW1ln6hzswAAApQAAAINcG9zdEkLMmU
AAASkAAAANgABAAADUv9qAFoEAP/z//4D6gABAAAAAAAAAAAAAAAAAAAAAgABAAAAAQAAAZfc6F8
PPPUACwPoAAAAANfSn9kAAAAA19Kf2f/z//wD6gPhAAAACAACAAAAAAAAAAEAAAACAD4AAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPhAJYAAAABAAAAAAAAABAAAAAPo//M
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAAZgAAAAL/8//8A+oD4QAKAD0AAAEWBgceATc1JiQHJTMmNjceARcVJx4BBx4BFQ4BIiYnNDY
3PgEvAS4BIw4BBwEGHgI3AT4BLwE1LgEnDgEDeiRlCgulCxP+8RT+GyYDQFxOZQwTBQEDDxEBJzo
nAREOCQkPJQ4cDBcdAf6oG1a3nx8BWQ4RHKADeG1oWwHTLHVwYVmL6Kx1BHEqfwYFqWUHEx4tDAo
cEx0nJx0RHgoVUDQpDgsBFAH+px2guFUaAVkNOiCgCXnhCAWOAAAAAAAAEgDeAAEAAAAAAAAAAQA
AAAEAAAAAAAEABQABAAEAAAAAAAIABwAGAAEAAAAAAAMABQANAAEAAAAAAAQABQASAAEAAAAAAAU
ACwAXAAEAAAAAAAYABQAiAAEAAAAAAAoALAAnAAEAAAAAAAsAEgBTAAMAAQQJAAAAAgBlAAMAAQQ
JAAEACgBnAAMAAQQJAAIADgBxAAMAAQQJAAMACgB/AAMAAQQJAAQACgCJAAMAAQQJAAUAFgCTAAM
AAQQJAAYACgCpAAMAAQQJAAoAWACzAAMAAQQJAAsAJAELIHBhaW50UmVndWxhcnBhaW50cGFpbnR
WZXJzaW9uIDEuMHBhaW50Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABwAGEAaQBuAHQAUgBlAGcAdQBsAGEAcgBwAGEAaQBuAHQ
AcABhAGkAbgB0AFYAZQByAHMAaQBvAG4AIAAxAC4AMABwAGEAaQBuAHQARgBvAG4AdAAgAGcAZQB
uAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHI
AbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMAbwBtAAAAAAI
AAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMADHBhaW50LWJ1Y2tldAAAAAA=)
format('truetype');
        font-weight: normal;
        font-style: normal;
    }
    .e-sb-icons {
        font-family: 'paint' !important;
        speak: none;
        font-size: 55px;
        font-style: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
```

```
        line-height: 1;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
    }
    .e-color::before {
    content: '\e700';
    color:black;
    }
    #dropdowntarget {
        border: 0.5px solid grey;
        height: 110px;
        width: 110px;
    }
    #black {
        width: 30px;
        height: 30px;
        background-color: black;
        margin: 5px 5px;
        float:left;
    }
    #red{
        width: 30px;
        height: 30px;
        background-color: red;
        margin: 5px 0px;
        float:left;
    }
    #green{
        width: 30px;
        height: 30px;
        background-color: green;
        margin: 5px 5px;
        float:left;
    }
    #gray{
        width: 30px;
        height: 30px;
        background-color: gray;
        margin: 0px 5px;
        float:left;
    }
    #blue{
        width: 30px;
        height: 30px;
        background-color: blue;
        float:left;
    }
    #violet{
        width: 30px;
        height: 30px;
        background-color: violet;
        margin: 0px 5px;
        float:left;
    }
    #brown{
        width: 30px;
        height: 30px;
```

```
        background-color: brown;
        margin: 5px 5px;
        float:left;
    }
    #darkgoldenrod{
        width: 30px;
        height: 30px;
        background-color: darkgoldenrod;
        margin: 5px 0px;
        float:left;
    }
    #aquamarine{
        width: 30px;
        height: 30px;
        background-color: aquamarine;
        margin: 5px 5px;
        float:left;
    }
    #icon{
        width: 10px;
        height: 10px;
        background-color: aquamarine;
        position: absolute;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/split-button/default-cs13" %}

See Also

- Popup items grouping
- SplitButton popup with separator

Accessibility in Vue Split button component

The Split button component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Split button component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Split button component followed the [WAI-ARIA] patterns to meet the accessibility. The following ARIA attributes are used in the Split button component:

| Attributes | Purpose |

| --- | --- |

| `role` | Indicates the Split button component as `button`, Split button popup as `menu`, and the Split button popup action items as `menuitem`. |

| `aria-haspopup` | Indicates the availability and type of interactive Split button popup element. |

| `aria-expanded` | Indicates whether the Split button popup can be expanded or collapsed, as well as indicates whether its current state is expanded or collapsed. |

| `aria-owns` | Identifies an elements in order to define a visual, functional, or contextual parent/child relationship between DOM elements where the DOM hierarchy cannot be used to represent the relationship. |

| `aria-disabled` | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Copyright © 2001 -2024 Syncfusion Inc.

Keyboard interaction

The Split button component followed the [keyboard interaction] guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Split button component.

| **Press** | **To do this** |

| --- | --- |

| Esc | Closes the opened popup. |

| Enter | Opens the popup, or activates the highlighted item and closes the popup. |

| Spacer | Opens the popup. |

| Up | Navigates up or to the previous action item. |

| Down | Navigates down or to the next action item. |

| Alt + Up Arrow | Closes the popup. |

| Alt + Down Arrow | Opens the popup. |

Ensuring accessibility

The Split button component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Split button component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Split button component with accessibility tools.

{% previewsample "page.domainurl/code-snippet/split-button/default-cs1" %}

See also

- Accessibility in Syncfusion Vue components

Style and appearance in Vue Split button component

To modify the SplitButton appearance, you need to override the default CSS of SplitButton component. Please find the list of CSS classes and its corresponding section in SplitButton. Also, you have an option to create your own custom theme for the controls using our Theme Studio.

CSS Class | Purpose of Class

|.e-dropdown-btn|To customize the button part in splitbutton

|.e-split-btn|To customize the dropdown part in split button

|.e-dropdown-popup ul .e-item|To customize the progress button list items

|.e-dropdown-popup ul .e-item .e-menu-icon |To customize the progress button list items icon

How To

Create right to left splitbutton in Vue Split button component

SplitButton component has RTL support. This can be achieved by setting enableRtl as `true`.

The following example illustrates how to enable right-to-left support in SplitButton component.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <ejs-splitbutton :items='items' iconCss='e-sb e-sigma'
enableRtl='true'>Autosum</ejs-splitbutton>
</template>
<script>
import Vue from 'vue';
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SplitButtonPlugin);
export default {
    data () {
        return {
            items:[
            {
                text: 'Autosum'
            },
            {
                text: 'Average'
            },
            {
                text: 'Count numbers',
            },
            {
                text: 'Min'
            },
            {
                text: 'Max'
            }]
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
  .e-split-btn-wrapper{
   margin: 20px 20px 5px 5px;
  }
    @font-face {
        font-family: 'sb-icon';
        src:
        url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj0gSRsAAAEoAAAAVmNtYXDnEOdVAAABiAAAADZnbHlmFjF
e0gAAAcgAAABIaGVhZBIxlUcAAADQAAAANmhoZWEHHANtAAAArAAAACRobXR4B+gAAAAAYAAAAA
IbG9jYQAkAAAAAHAAAAABm1heHABDQAeAAABCAAAACBuYW1lakQFAwAAAhAAAAIlcG9zdEP61+c
AAAQ4AAAAMwABAAADUv9qAFoEAAAAAADbgABAAAAAAAAAAAAAAAAAAAAAgABAAAAAQAAj4iO918
PPPUACwPoAAAAANfSqDwAAAAA19KoPAAAAAADbgPqAAAACAACAAAAAAAAAAEAAAACABIAAQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPqAJYAAAABAAAAAAAABAAAAAPoAAA
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAAJAAAAAEAAAAAA24D6gARAAATHQETARUhEyMHIQEDIRUzESG65f7gAsolLBH9ywES9gIFLf1
wA5xDQv6u/pBSAQNyAVwBaXIBBAAAAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAEABwABAAEAAAA
```

```
AAAIABwAIAAEAAAAAAAMABwAPAAEAAAAAAAQABwAWAAEAAAAAAAUACwAdAAEAAAAAAAYABwAoAAE
AAAAAAAoALAAvAAEAAAAAAAsAEgBbAAMAAQQJAAAAAgBtAAMAAQQJAAEADgBvAAMAAQQJAAIADgB
9AAMAAQQJAAMADgCLAAMAAQQJAAQADgCZAAMAAQQJAAUAFgCnAAMAAQQJAAYADgC9AAMAAQQJAAo
AWADLAAMAAQQJAAsAJAEjIHNiLWljb25SZWd1bGFyc2ItaWNvbnNiLWljb25WZXJzaW9uIDEuMHN
iLWljb25Gb250IGdlbmVyYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAHMAYgAtAGkAYwBvAG4AUgBlAGcAdQBsAGEAcgBzAGIALQBpAGMAbwBuAHM
AYgAtAGkAYwBvAG4AVgBlAHIAcwBpAG8AbgAgADEALgAwAHMAYgAtAGkAYwBvAG4ARgBvAG4AdAA
gAGcAZQBuAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0
AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMAbwB
tAAAAAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMACXN1bW1hdGlvbgAAAA=
=) format('truetype');
        font-weight: normal;
        font-style: normal;
    }
    .e-sb {
        font-family: 'sb-icon' !important;
        speak: none;
        font-size: 55px;
        font-style: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
        line-height: 1;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
    }
    .e-sigma::before {
        content: '\e700';
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/split-button/default-cs3" %}

Group items in popup in Vue Split button component

Grouped items are possible in SplitButton by templating entire popup with ListView. Check ListView grouping and create such items. Create ListView with id listview and provide element of the ListView as target of SplitButton to render it in popup area.

Install Syncfusion List packages using below command.

`bash

npm install @syncfusion/ej2-vue-list --save

`

In this following example, ListView is created and its element is set as target for SplitButton.

APP.VUE

```
<template>
<div>
    <ejs-splitbutton target='#listview'>ClipBoard</ejs-splitbutton>
    <ejs-listview id='listview' :dataSource='items' :fields='fields'
sortOrder='Descending'>
    </ejs-listview>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { ListViewPlugin } from '@syncfusion/ej2-vue-lists';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SplitButtonPlugin);
Vue.use(ListViewPlugin);
export default {
    data () {
        return {
            items:[
            {
                'text': 'Cut',
                'category': 'Basic'
            },
            {
                'text': 'Copy',
                'category': 'Basic'
            },
            {
                'text': 'Paste',
                'category': 'Basic'
            },
            {
                'text': 'Paste as Formula',
                'category': 'Advanced'
            },
            {
                'text': 'Paste as Hyperlink',
                'category': 'Advanced'
            }],
            fields: { groupBy: 'category' }
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
  .e-split-btn-wrapper{
   margin: 20px 20px 5px 5px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/split-button/default-cs4" %}

Open a dialog on popup item click in Vue Split button component

This section explains about how to open a dialog on SplitButton popup item click. This can be achieved by handling dialog open in select event of the SplitButton.

Install Syncfusion Popup packages using below command.

Copyright © 2001 -2024 Syncfusion Inc.

`bash

npm install @syncfusion/ej2-vue-popups --save

`

In the following example, Dialog will open while selecting Update... item:

APP.VUE

```
<template>
<div>
    <ejs-splitbutton :items='items' :select='onSelect'>Help</ejs-
splitbutton>
    <ejs-dialog id="dialog" content="Are you sure want to update?"
header='Software Update' :buttons='buttons' width='250px' :visible='false'
></ejs-dialog>
</div>
</template>
<script>
import Vue from 'vue';
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { DialogPlugin } from "@syncfusion/ej2-vue-popups";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SplitButtonPlugin);
Vue.use(DialogPlugin);
export default {
    data () {
        return {
            items:[
            {
                text: 'Help'
            },
            {
                text: 'About'
            },
            {
                text: 'Update...'
            }],
            buttons: [{
                buttonModel: {
                    isPrimary: true,
                    content: 'Ok',
                },
                click: function () {
                    this.hide();
                }
            },
            {
                buttonModel: {
                    isPrimary: true,
                    content: 'Cancel',
                },
                click: function () {
                    this.hide();
                }
            }]
```

```
        };
    },
    methods: {
        onSelect: function(args) {
            if (args.item.text === 'Update...') {
                document.getElementById("dialog").ej2_instances[0].show();
            }
        }
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
  .e-split-btn-wrapper{
   margin: 20px 20px 5px 5px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/split-button/default-cs5" %}

Set the disabled state in Vue Split button component

SplitButton component can be enabled or disabled by disabled property. To disable SplitButton component, set the disabled property as true.

The following example illustrates how to set the disable state in SplitButton component.

APP.VUE

```
<template>
    <ejs-splitbutton :items='items' iconCss='e-sb e-sigma'
disabled='true'>Autosum</ejs-splitbutton>
</template>
<script>
import Vue from 'vue';
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SplitButtonPlugin);
export default {
    data () {
        return {
            items:[
            {
                text: 'Autosum'
            },
            {
                text: 'Average'
            },
            {
                text: 'Count numbers',
            },
            {
                text: 'Min'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            },
            {
                text: 'Max'
            }]
        };
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
  .e-split-btn-wrapper{
   margin: 20px 20px 5px 5px;
  }
    @font-face {
        font-family: 'sb-icon';
        src:
        url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj0gSRsAAAEoAAAAVmNtYXDnEOdVAAABiAAAADZnbHlmFjF
e0gAAAcgAAABIaGVhZBIxlUcAAADQAAAANmhoZWEHHANtAAAArAAAACRobXR4B+gAAAAAYAAAAA
IbG9jYQAkAAAAAAHAAAAABm1heHABDQAeAAABCAAAACBuYW1lakQFAwAAAhAAAAIlcG9zdEP61+c
AAAQ4AAAAMwABAAADUv9qAFoEAAAAAADbgABAAAAAAAAAAAAAAAAAAAAAAAgABAAAAAQAAj4iO918
PPPUACwPoAAAAANfSqDwAAAAA19KoPAAAAAADbgPqAAAACAACAAAAAAAAAAEAAAACABIAAQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPqAJYAAAABAAAAAAAABAAAAAPoAAA
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAAJAAAAAEAAAAA24D6gARAAATHQETARUhEyMHIQEDIRUzESG65f7gAsolLBH9ywES9gIFLf1
wA5xDQv6u/pBSAQNyAVwBaXIBBAAAAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAAEABwABAAEAAAA
AAAIABwAIAAEAAAAAAAMABwAPAAEAAAAAAAAQABwAWAAEAAAAAAAAUACwAdAAEAAAAAAAAYABwAoAAE
AAAAAAAoALAAvAAEAAAAAAAAsAEgBbAAMAAQQJAAAAAgBtAAMAAQQJAAEADgBvAAMAAQQJAAIADgB
9AAMAAQQJAAMADgCLAAMAAQQJAAQADgCZAAMAAQQJAAUAFgCnAAMAAQQJAAYADgC9AAMAAQQJAAo
AWADLAAMAAQQJAAsAJAEjIHNiLWljb25SZWd1bGFyc2ItaWNvbnNiLWljb25WZXJzaW9uIDEuMHN
iLWljb25Gb250IGdlbmVyYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAHMAYgAtAGkAYwBvAG4AUgBlAGcAdQBsAGEAcgBzAGIALQBpAGMAbwBuAHM
AYgAtAGkAYwBvAG4AVgBlAHIAcwBpAG8AbgAgADEALgAwAHMAYgAtAGkAYwBvAG4ARgBvAG4AdAA
gAGcAZQBuAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0
AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMAbwB
tAAAAAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMACXN1bW1hdGlvbgAAAA=
=) format('truetype');
        font-weight: normal;
        font-style: normal;
    }
    .e-sb {
        font-family: 'sb-icon' !important;
        speak: none;
        font-size: 55px;
        font-style: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
        line-height: 1;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
    }
    .e-sigma::before {
```

```
        content: '\e700';
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/split-button/default-cs6" %}

Underline a character in a text in Vue Split button component

To underline a particular character in a text, it can be handled in beforeItemRender event by adding <u> tag in between the text and given as innerHTML in li rendering.

In the following example, C is underlined in the text Copy:

APP.VUE

```
<template>
    <ejs-splitbutton :items='items'
:beforeItemRender='itemRender'>Paste</ejs-splitbutton>
</template>
<script>
import Vue from 'vue';
import { SplitButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SplitButtonPlugin);
export default {
    data () {
        return {
            items:[
            {
                text: 'Cut'
            },
            {
                text: 'Copy'
            },
            {
                text: 'Paste'
            }]
        };
    },
    methods: {
        itemRender: function(args) {
            if (args.item.text === 'Copy') {
                //To underline a particular text.
                args.element.innerHTML = '<u>C</u>opy';
            }
        }
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
  .e-split-btn-wrapper{
   margin: 20px 20px 5px 5px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/split-button/default-cs7" %}

Splitter

Getting Started with the Vue Splitter Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Splitter component.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies required to use the Splitter component in your application:

`js

|-- @syncfusion/ej2-layouts

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-vue-layouts

|-- @syncfusion/ej2-vue-base

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.

```
E:\samples>vue create quickstart


Vue CLI v5.0.8
? Please pick a preset:
  Default ([Vue 3] babel, eslint)
> Default ([Vue 2] babel, eslint)
  Manually select features
```

Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Splitter component as an example. Install the @syncfusion/ej2-vue-layouts package by running the following command:

`bash

npm install @syncfusion/ej2-vue-layouts --save

`

or

`bash

yarn add @syncfusion/ej2-vue-layouts

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Splitter component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css';
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Splitter component using:

1\. First, import and register the Splitter component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { SplitterComponent, PanesDirective, PaneDirective } from
'@syncfusion/ej2-vue-layouts';
export default {
components: {
'ejs-splitter': SplitterComponent,
'e-pane': PaneDirective,
'e-panes': PanesDirective
}
}
</script>
```

2\. In the template section, define the Splitter component with the height property.

~/SRC/APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
<ejs-splitter id='default-splitter' height='250px'>
<e-panes>
<e-pane></e-pane>
<e-pane></e-pane>
</e-panes>
</ejs-splitter>
</div>
</template>
```

Load content to the pane

You can load the pane contents in either HTML element or string types using content property.

For detailed information, refer to the Pane Content section

~/SRC/APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
<ejs-splitter id='default-splitter' height='250px'>
<e-panes>
<e-pane content='Left Pane'></e-pane>
<e-pane content='Middle Pane'></e-pane>
<e-pane content='Right Pane'></e-pane>
</e-panes>
</ejs-splitter>
</div>
</template>
<script>
import { SplitterComponent, PanesDirective, PaneDirective } from
'@syncfusion/ej2-vue-layouts';
export default {
components: {
'ejs-splitter': SplitterComponent,
'e-pane': PaneDirective,
'e-panes': PanesDirective
},
name: 'app',
data () {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
return { }
}
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

~/SRC/APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='default-splitter' height='250px'>
        <e-panes>
            <e-pane content='Left Pane'></e-pane>
            <e-pane content='Middle Pane'></e-pane>
            <e-pane content='Right Pane'></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import { SplitterComponent, PanesDirective, PaneDirective } from
'@syncfusion/ej2-vue-layouts';
export default {
  components: {
    'ejs-splitter': SplitterComponent,
    'e-pane': PaneDirective,
    'e-panes': PanesDirective
  },
  name: 'app',
    data () {
    return { }
  }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css';
#app {
    margin: 60px auto;
    max-width: 700px;
}
.e-splitter {
    margin: 0 auto;
}
.e-pane {
  text-align: center;
  align-items: center;
  display: grid;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/splitter/getting-started-cs1" %}

See Also

- Resizable split panes
- Collapsible panes
- Construct different layouts using Splitter

Getting Started with the Vue Splitter Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Splitter component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

Copyright © 2001 -2024 Syncfusion Inc.

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

Copyright © 2001 -2024 Syncfusion Inc.

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Splitter component as an example. To use the Vue Splitter component in the project, the @syncfusion/ej2-vue-layouts package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-layouts --save

`

or

`bash

yarn add @syncfusion/ej2-vue-layouts

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Splitter component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css';
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue Splitter component using Composition API or Options API:

1.First, import and register the Splitter component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { SplitterComponent as EjsSplitter, PanesDirective as EPanes,
PaneDirective as Epane } from "@syncfusion/ej2-vue-layouts";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SplitterComponent, PanesDirective, PaneDirective } from
"@syncfusion/ej2-vue-layouts";
//Component registration
export default {
name: "App",
components: {
'ejs-splitter' : SplitterComponent,
'e-panes': PanesDirective,
'e-pane': PaneDirective
},
}
</script>
```

2.In the template section, define the Splitter component with the dataSource property and column definitions.

~/SRC/APP.VUE

```
<template>
<ejs-splitter id='default-splitter' height='250px'>
<e-panes>
<e-pane content='Left Pane'></e-pane>
<e-pane content='Middle Pane'></e-pane>
<e-pane content='Right Pane'></e-pane>
</e-panes>
</ejs-splitter>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-splitter id='default-splitter' height='250px'>
<e-panes>
<e-pane content='Left Pane'></e-pane>
<e-pane content='Middle Pane'></e-pane>
<e-pane content='Right Pane'></e-pane>
</e-panes>
</ejs-splitter>
</template>
<script setup>
import { SplitterComponent as EjsSplitter, PanesDirective as EPanes,
PaneDirective as Epane } from "@syncfusion/ej2-vue-layouts";
const data = [];
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css';
#app {
margin: 60px auto;
max-width: 700px;
}
.e-splitter {
margin: 0 auto;
}
.e-pane {
text-align: center;
align-items: center;
display: grid;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-splitter id='default-splitter' height='250px'>
<e-panes>
<e-pane content='Left Pane'></e-pane>
<e-pane content='Middle Pane'></e-pane>
<e-pane content='Right Pane'></e-pane>
</e-panes>
</ejs-splitter>
</template>
<script>
import { SplitterComponent, PanesDirective, PaneDirective } from
"@syncfusion/ej2-vue-layouts";
export default {
name: 'App',
components: {
'ejs-splitter' : SplitterComponent,
'e-panes': PanesDirective,
'e-pane': PaneDirective
},
data() {
return {
};
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css';
#app {
margin: 60px auto;
max-width: 700px;
}
.e-splitter {
margin: 0 auto;
}
.e-pane {
text-align: center;
align-items: center;
display: grid;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

Pane sizing in Vue Splitter component

Splitter allows you to provide pane sizes either in pixel or percentage formats.

Auto size panes

The splitter's panes are adjusted automatically during resizing if the size is not specified externally to panes, because the panes are designed based on flex layout by default. When add/remove or show/hide the panes, the panes are auto aligned within its container.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='600px'
height='200px'>
        <e-panes>
            <e-pane :content='content1'></e-pane>
            <e-pane :content='content2'></e-pane>
            <e-pane :content='content3'></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
var contentVue1 = Vue.component("contentTemp1", {
    template: `<div class="content">
        <h3 class="h3">Grid </h3>
        The ASP.NET DataGrid component, or DataTable is a feature-rich
component used to display data in a tabular format.
    </div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue2 = Vue.component("contentTemp2", {
    template: `<div class="content">
        <h3 class="h3">Schedule </h3>
        The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all
calendar features, thus allowing users to manage their time efficiently
    </div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue3 = Vue.component("contentTemp3", {
    template: `<div class="content">
        <h3 class="h3">Chart </h3>
        ASP.NET charts, a well-crafted easy-to-use charting package, is used
to add beautiful charts in web and mobile applications
    </div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
    name: 'app',
    data () {
        return {
        content1: function (){
            return { template: contentVue1 }
        },
        content2: function (){
            return { template: contentVue2 }
        },
        content3: function (){
            return { template: contentVue3 }
        }
        }
    }
}
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    margin: 65px auto;
}
.content {
    padding: 10px;
}
.e-splitter {
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/resize-min-max-cs4" %}

Fixed pane

You can render the split panes with fixed size in both horizontal and vertical mode. Even if you provide fixed sizes to all the panes, the last pane is considered as a flexible pane.

Last pane is flexible because the splitter needs one pane as flexible pane always, to adjust its remaining layout space.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='600px'
height='200px'>
        <e-panes>
            <e-pane :content='content1' size ='200px'></e-pane>
            <e-pane :content='content2' size ='200px'></e-pane>
            <e-pane :content='content3' size ='200px'></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
var contentVue1 = Vue.component("contentTemp1", {
    template: `<div>Left pane</div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue2 = Vue.component("contentTemp2", {
    template: `<div>Middle pane</div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue3 = Vue.component("contentTemp3", {
```

```
    template: `<div>Right pane</div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
    name: 'app',
    data () {
        return {
        content1: function (){
            return { template: contentVue1 }
        },
        content2: function (){
            return { template: contentVue2 }
        },
        content3: function (){
            return { template: contentVue3 }
        }
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    text-align: center;
    margin: 65px auto;
}
.content {
  justify-content: center;
  text-align: center;
  align-items: center;
}
.e-splitter {
    margin: 0 auto;
}
.e-pane {
    align-items: center;
    display: grid;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/resize-min-max-cs5" %}

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='600px'
height='200px'>
        <e-panes>
            <e-pane :content='content1' size ='30%'></e-pane>
            <e-pane :content='content2' size ='40%'></e-pane>
            <e-pane :content='content3' size ='30%'></e-pane>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
var contentVue1 = Vue.component("contentTemp1", {
    template: `<div>Left pane</div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue2 = Vue.component("contentTemp2", {
    template: `<div>Middle pane</div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue3 = Vue.component("contentTemp3", {
    template: `<div>Right pane</div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
    name: 'app',
    data () {
        return {
        content1: function (){
            return { template: contentVue1 }
        },
        content2: function (){
            return { template: contentVue2 }
        },
        content3: function (){
            return { template: contentVue3 }
        }
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    text-align: center;
    margin: 65px auto;
}
.content {
  justify-content: center;
```

```
  text-align: center;
  align-items: center;
}
.e-splitter {
    margin: 0 auto;
}
.e-pane {
    align-items: center;
    display: grid;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/resize-min-max-cs6" %}

Pane content in Vue Splitter component

This section explains how to provide plain text content or HTML markup or integrate other JavaScript UI components as content of splitter.

HTML Markup

Splitter is a layout based container component. You can render the pane contents from existing HTML markups. Converting HTML markup as splitter pane is easy way to add the panes content dynamically.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='600px'
height='200px'>
        <e-panes>
            <e-pane :content='content1' size = '200px'></e-pane>
            <e-pane :content='content2' size = '200px'></e-pane>
            <e-pane :content='content3' size = '200px'></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
var contentVue1 = Vue.component("contentTemp1", {
    template: `<div class="content"><h3 class="h3">PARIS </h3>Paris, the
city of lights and love - this short guide is full of ideas for how to make
the most of the romanticism...</div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue2 = Vue.component("contentTemp2", {
    template: `<div class="content"><h3 class="h3">CAMEMBERT </h3>The
village in the Orne département of Normandy where the famous French cheese
is originated from.</div>`,
    data() {
        return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            data: {}
        };
    }
});
var contentVue3 = Vue.component("contentTemp3", {
    template: `<div class="content"><h3 class="h3">GRENOBLE </h3>The capital
city of the French Alps and a major scientific center surrounded by many ski
resorts, host of the Winter Olympics in 1968.</div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
    name: 'app',
    data () {
        return {
        content1: function (){
            return { template: contentVue1 }
        },
        content2: function (){
            return { template: contentVue2 }
        },
        content3: function (){
            return { template: contentVue3 }
        }
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    margin: 65px auto;
}
.content {
    padding: 10px;
}
.e-splitter {
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/resize-min-max-cs1" %}

JavaScript UI components

You can render any JavaScript UI components along with their native and control events within splitter as pane content.

You can refer Accordion within splitter and Listview within splitter examples.

Plain content

You can add the plain text as a pane contents using either inner HTML or content API

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='600px'
height='200px'>
        <e-panes>
            <e-pane content='Left Pane' size ='200px'></e-pane>
            <e-pane content='Middle Pane' size ='200px'></e-pane>
            <e-pane content='Right Pane' size ='200px'></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
export default {
    name: 'app',
    data () {
        return { }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    text-align: center;
    margin: 65px auto;
}
.content {
  justify-content: center;
  text-align: center;
  align-items: center;
  padding: 9px;
}
.e-splitter {
    margin: 0 auto;
}
.e-pane {
  text-align: center;
  align-items: center;
  display: grid;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/resize-min-max-cs2" %}

Pane content using selector

You can set HTML element as pane content using the query selectors such as ID or CSS class selectors.

The following code demonstrates how to fetch an element from the document and load it to pane using its ID.

APP.VUE

```
<template>
```

```
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj' width='600px'
height='200px'>
        <e-panes>
            <e-pane content='#left-pane-content' min='60px' size
='200px'></e-pane>
            <e-pane content='#middle-pane-content'  min='60px' size
='200px'></e-pane>
            <e-pane content='#last-pane-content'  min='60px' size
='200px'></e-pane>
        </e-panes>
    </ejs-splitter>
    <!-- pane contents -->
    <div id="left-pane-content" style="display: none;">
        <div>Left pane<div id='panetext'>size: 25%</div>
            <div id='panetext'>min: 60px</div>
        </div>
    </div>
    <div id="middle-pane-content" style="display: none;">
        <span>Middle pane<div id="panetext">size: 50%</div>
            <div id="panetext">min: 60px</div>
        </span>
    </div>
    <div id="last-pane-content" style="display: none;">
        <span>Right pane<div id="panetext">size: 25%</div>
            <div id="panetext">min: 60px</div>
        </span>
    </div>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
export default {
    name: 'app',
    data () {
        return { }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    text-align: center;
    margin: 65px auto;
}
.content {
  justify-content: center;
  text-align: center;
  align-items: center;
  padding: 9px;
}
.e-splitter {
    margin: 0 auto;
}
.e-pane {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  text-align: center;
  align-items: center;
  display: grid;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/resize-min-max-cs3" %}

Split panes in Vue Splitter component

This section explain about split panes behaviours.

Horizontal layout

By default, splitter will render in horizontal orientation. Splitter container will be splitted as panes in horizontal flow direction with vertical separator.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='default-splitter' width='600px' height='230px'>
        <e-panes>
            <e-pane :content ='content1' size ='200px'></e-pane>
            <e-pane :content ='content2' size ='200px'></e-pane>
            <e-pane :content ='content3' size ='200px'></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
var contentVue1 = Vue.component("contentTemp1", {
    template: `<div class="content">
                    <h3 class="h3">Grid </h3>
                    The ASP.NET DataGrid component, or DataTable is a
feature-rich component used to display data in a tabular format.
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue2 = Vue.component("contentTemp2", {
    template: `<div class="content">
                    <h3 class="h3">Schedule </h3>
                    The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
                </div>`,
    data() {
        return {
            data: {}
        };
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
});
var contentVue3 = Vue.component("contentTemp3", {
    template: `<div class="content">
                    <h3 class="h3">Chart </h3>
                    ASP.NET charts, a well-crafted easy-to-use charting
package, is used to add beautiful charts in web and mobile applications
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
  name: 'app',
    data () {
    return {
    content1: function (){
        return { template: contentVue1 }
    },
    content2: function (){
        return { template: contentVue2 }
    },
    content3: function (){
        return { template: contentVue3 }
    }
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    margin: 65px auto;
}
.content {
    padding: 10px;
}
.e-splitter {
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/split-panes-cs1" %}

Vertical layout

By setting orientation API as Vertical, splitter will render in vertical orientation. Splitter container will be splitted as panes in vertical flow direction with horizontal separator.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='vertical-splitter' orientation = 'Vertical'
width='600px' height='300px'>
        <e-panes>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <e-pane :content ='content1' size ='100px'></e-pane>
            <e-pane :content ='content2' size ='100px'></e-pane>
            <e-pane :content ='content3'></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
var contentVue1 = Vue.component("contentTemp1", {
    template: `<div class="content">
                    <h3 class="h3">Grid</h3>
                    The ASP.NET DataGrid component, or DataTable is a
feature-rich component used to display data in a tabular format.
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue2 = Vue.component("contentTemp2", {
    template: `<div class="content">
                    <h3 class="h3">Schedule </h3>
                    The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue3 = Vue.component("contentTemp3", {
    template: `<div class="content">
                    <h3 class="h3">Chart </h3>
                    ASP.NET charts, a well-crafted easy-to-use charting
package, is used to add beautiful charts in web and mobile applications
              </div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
  name: 'app',
    data () {
    return {
    content1: function (){
        return { template: contentVue1 }
    },
    content2: function (){
        return { template: contentVue2 }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    },
    content3: function (){
        return { template: contentVue3 }
    }
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
.content {
  padding: 8px;
}
.e-splitter {
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/panes-orientation-cs1" %}

You can also render multiple panes in splitter with both Horizontal/Vertical orientations.

Separator

By default, pane separator will be render with 1px width/height. You can customize the separator size by using separatorSize API.

For horizontal orientation, it will be considered as separator width.

For vertical orientation, it will be considered as separator height.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' width='600px' height='235px' separatorSize=
'5'>
        <e-panes>
            <e-pane :content ='content1' size ='200px'></e-pane>
            <e-pane :content ='content2' size ='200px'></e-pane>
            <e-pane :content ='content3' size ='200px'></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
var contentVue1 = Vue.component("contentTemp1", {
    template: `<div class="content">
                    <h3 class="h3">PARIS </h3>
                    Paris, the city of lights and love - this short guide is
full of ideas for how to make the most of the romanticism...
                </div>`,
    data() {
        return {
            data: {}
```

```
        };
    }
});
var contentVue2 = Vue.component("contentTemp2", {
    template: `<div class="content">
                    <h3 class="h3">CAMEMBERT </h3>
                    The village in the Orne département of Normandy where
the famous French cheese is originated from.
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue3 = Vue.component("contentTemp3", {
    template: `<div class="content">
                    <h3 class="h3">GRENOBLE </h3>
                    The capital city of the French Alps and a major
scientific center surrounded by many ski resorts, host of the Winter
Olympics in 1968.
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
  name: 'app',
    data () {
    return {
    content1: function (){
        return { template: contentVue1 }
    },
    content2: function (){
        return { template: contentVue2 }
    },
    content3: function (){
        return { template: contentVue3 }
    }
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    margin: 65px auto;
}
.content {
    padding: 10px;
}
.e-splitter {
    margin: 0 auto;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/splitter/panes-separator-cs1" %}

Nested Splitter

Splitter provides support to render the nested pane to achieve the complex layouts. You can use the same `<div>` element for splitter pane and nested splitter.

Also you can render the nested splitter using direct child of the splitter pane. For this, nested splitter should have `100%` width and height to match with the parent pane dimensions.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='nested-splitter' width='600px' height='265px'
:created='onCreate' >
        <e-panes>
            <e-pane :content ='content1' size ='200px'></e-pane>
            <e-pane :content ='content2' size ='200px'></e-pane>
            <e-pane :content ='content3' size ='200px'></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
import {Splitter} from '@syncfusion/ej2-layouts';
Vue.use(SplitterPlugin);
var contentVue1 = Vue.component("contentTemp1", {
    template: `<div class="content">
                    <h3 class="h3">PARIS </h3>
                    Paris, the city of lights and love - this short guide is
full of ideas for how to make the most of the romanticism...
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue2 = Vue.component("contentTemp2", {
    template: `<div class="content">
                    <h3 class="h3">CAMEMBERT </h3>
                    The village in the Orne département of Normandy where
the famous French cheese is originated from.
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue3 = Vue.component("contentTemp3", {
    template: `<div>
                <div id ='vertical_splitter' class="vertical_splitter">
```

```
                    <div>
                        <div class="content">
                            <h3 class="h3">GRENOBLE </h3>
                            The capital city of the French Alps and a major
scientific center surrounded by many ski resorts, host of the Winter
Olympics in 1968.
                        </div>
                    </div>
                    <div>
                        <div class="content">
                            <h3 class="h3">Australia </h3>
                            Australia is a country and continent surrounded
by the Indian and Pacific oceans
                        </div>
                    </div>
                </div>
            </div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
  name: 'app',
    data () {
    return {
    content1: function (){
        return { template: contentVue1 }
    },
    content2: function (){
        return { template: contentVue2 }
    },
    content3: function (){
        return { template: contentVue3 }
    }
    }
  },
  methods: {
    onCreate: function () {
        this.splitterObj = new Splitter({
            paneSettings: [
                { size: '150px', min: '20%' },
                { size: '100px', min: '20%'}
            ],
            orientation: 'Vertical'
        });
        this.splitterObj.appendTo('#vertical_splitter');
    }
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    margin: 55px auto;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.content {
    padding: 5px;
}
.vertical_splitter.e-splitter.e-splitter-vertical  {
  border: none;
}
.e-splitter {
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/nested-splitter-cs1" %}

Add or remove pane

You can add or remove panes programmatically to the splitter, by using addPane and removePane methods.

Add pane

You can add the panes dynamically in the splitter by passing paneProperties along with index to the addPane method

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='700px'
height='200px'>
        <e-panes>
            <e-pane content ='Pane 1' size ='300px'></e-pane>
            <e-pane content ='Pane 2' size ='300px'></e-pane>
        </e-panes>
    </ejs-splitter>
    <div class="addBtn">
      <button id='addButton' class="e-btn" v-on:click = 'addPane'>Add
Pane</button>
    </div>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
import { PanePropertiesModel } from '@syncfusion/ej2-layouts';
Vue.use(SplitterPlugin);
export default {
  name: 'app',
    data () {
    return {
    }
  },
  methods: {
    addPane: function () {
         this.$refs.splitterObj.$el.ej2_instances[0].addPane({
      size: '200px',
      content: 'New Pane',
      min: '30px',
      max: '250px',
```

```
      }, 1);
    }
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    margin: 65px auto;
}
#splitter1 div.e-pane-horizontal {
    padding: 5px;
}
.addBtn {
  margin-top: 20px;
  text-align: initial;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/add-pane-cs1" %}

Remove pane

You can remove the split panes dynamically by passing the pane index to removePane method.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='700px'
height='200px'>
        <e-panes>
            <e-pane content ='Pane 1' size ='300px'></e-pane>
            <e-pane content ='Pane 2' size ='300px'></e-pane>
            <e-pane content ='Pane 3' size ='300px'></e-pane>
        </e-panes>
    </ejs-splitter>
    <div class="removeBtn">
      <button id='removeButton' class="e-btn" v-on:click =
'removePane'>Remove Pane</button>
    </div>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
export default {
    name: 'app',
        data () {
        return {
        }
    },
    methods: {
        removePane: function () {
            this.$refs.splitterObj.removePane(1);
        }
```

```
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    text-align: center;
    margin: 65px auto;
}
.content {
  justify-content: center;
  text-align: center;
  align-items: center;
}
.removeBtn {
  margin-top: 20px;
  text-align: initial;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/remove-pane-cs1" %}

See Also

- Resizable split panes
- Collapsible panes
- Define size to a panes
- Specify content to a panes

Expand collapse in Vue Splitter component

Collapsible panes

The Vue Splitter panes can be configured with built-in expand and collapse functionalities. By default, the collapsible behavior is disabled. Enable the collapsible behavior in the paneSettings property to show or hide the expand or collapse icons in the panes. You can dynamically expand and collapse the panes by using the corresponding icons.

The following code shows how to enable collapsible behavior.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='600px'
height='200px'>
        <e-panes>
            <e-pane :content='content1' size ='200px' :collapsible=true></e-
pane>
            <e-pane :content='content2' size ='200px' :collapsible=true></e-
pane>
            <e-pane :content='content3' size ='200px' :collapsible=true></e-
pane>
        </e-panes>
    </ejs-splitter>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
var contentVue1 = Vue.component("contentTemp1", {
    template: `<div class="content"><h3 class="h3">PARIS </h3>Paris, the
city of lights and love - this short guide is full of ideas for how to make
the most of the romanticism...</div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue2 = Vue.component("contentTemp2", {
    template: `<div class="content"><h3 class="h3">CAMEMBERT </h3>The
village in the Orne département of Normandy where the famous French cheese
is originated from.</div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue3 = Vue.component("contentTemp3", {
    template: `<div class="content"><h3 class="h3">GRENOBLE </h3>The capital
city of the French Alps and a major scientific center surrounded by many ski
resorts, host of the Winter Olympics in 1968.</div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
    name: 'app',
    data () {
        return {
        content1: function (){
            return { template: contentVue1 }
        },
        content2: function (){
            return { template: contentVue2 }
        },
        content3: function (){
            return { template: contentVue3 }
        }
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    margin: 65px auto;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.content {
  padding: 10px;
}
.e-splitter {
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/expand-collapse-cs1" %}

Programmatically control the expand and collapse action

The Vue Splitter provides public method to control the expand and collapse behavior programmatically using the expand and collapse methods. Refer to the following code example.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='600px'
height='200px'>
        <e-panes>
            <e-pane :collapsible=true content='Left Pane' size ='200px'></e-
pane>
            <e-pane :collapsible=true content='Middle Pane' size
='200px'></e-pane>
            <e-pane :collapsible=true content='Right Pane' size
='200px'></e-pane>
        </e-panes>
    </ejs-splitter>
    <div style="margin: 10px">
        <ejs-button id='expand' v-on:click.native="expandClick">Expand</ejs-
button>
        <ejs-button id='collapse' v-
on:click.native="collapseClick">Collapse</ejs-button>
    </div>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(SplitterPlugin);
Vue.use(ButtonPlugin);
export default {
    name: 'app',
    data () {
        return { }
    },
    methods: {
        expandClick : function() {
            this.$refs.splitterObj.expand(0);
        },
        collapseClick : function() {
            this.$refs.splitterObj.collapse(0);
        }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
#app {
    text-align: center;
    margin: 45px auto;
}
.content {
  justify-content: center;
  text-align: center;
  align-items: center;
  padding: 9px;
}
.e-splitter {
    margin: 0 auto;
}
.e-pane {
  text-align: center;
  align-items: center;
  display: grid;
}
#collapse {
  margin-left: 10px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/expand-collapse-method-cs1" %}

Specify initial state to panes

You can render specific panes with collapsed state on page load. Specify a Boolean value to the collapsed property to control this behavior. The following code explains how to collapse panes on rendering (the second panes renders with collapsed state).

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='600px'
height='200px'>
        <e-panes>
            <e-pane content='Left Pane' size ='200px' :collapsible=true></e-
pane>
            <e-pane :collapsed=true content='Middle Pane' size ='200px'
:collapsible=true></e-pane>
            <e-pane content='Right Pane' size ='200px'
:collapsible=true></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
    name: 'app',
    data () {
        return { }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    text-align: center;
    margin: 65px auto;
}
.content {
  justify-content: center;
  text-align: center;
  align-items: center;
  padding: 9px;
}
.e-splitter {
    margin: 0 auto;
}
.e-pane {
  text-align: center;
  align-items: center;
  display: grid;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/collapsed-cs1" %}

See Also

- Resizable split panes

Resizing in Vue Splitter component

By default, resizing will be enable for split panes. Resizing gripper element will be add to the separator to makes the resize easy.

Horizontal splitter will allows to resize in horizontal directions.

Vertical splitter will allows to resize in vertical directions.

While resizing, previous and next panes will be adjust its dimensions automatically.

Min and Max validation

Splitter allows you to set the minimum and maximum sizes for each pane. Resizing will not be occur over the minimum and maximum values.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='600px'
height='200px'>
        <e-panes>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <e-pane :content='content1' size ='200px' min='20%'
max='40%'></e-pane>
            <e-pane :content='content2' size ='200px' min='30%'
max='60%'></e-pane>
            <e-pane :content='content3' size ='200px'></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
var contentVue1 = Vue.component("contentTemp1", {
    template: `<div class="content">
                    <h3 class="h3">Grid </h3>
                    The ASP.NET DataGrid component, or DataTable is a
feature-rich component used to display data in a tabular format.
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue2 = Vue.component("contentTemp2", {
    template: `<div class="content">
                    <h3 class="h3">Schedule </h3>
                    The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue3 = Vue.component("contentTemp3", {
    template: `<div class="content">
                    <h3 class="h3">Chart </h3>
                    ASP.NET charts, a well-crafted easy-to-use charting
package, is used to add beautiful charts in web and mobile applications
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
    name: 'app',
    data () {
        return {
        content1: function (){
            return { template: contentVue1 }
        },
```

```
        content2: function (){
            return { template: contentVue2 }
        },
        content3: function (){
            return { template: contentVue3 }
        }
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    margin: 65px auto;
}
.content {
  padding: 10px;
}
.e-splitter {
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/resize-min-max-cs7" %}

Prevent resizing

You can disable the resizing for the pane by setting `false` to the resizable API within paneSettings.

Splitter resizing will be enabled only when the target of the adjacent pane's `resizable` api should also be in `true` state.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='600px'
height='200px'>
        <e-panes>
            <e-pane :content='content1' size ='200px' min='20%' max='40%'
:resizable ='resize'></e-pane>
            <e-pane :content='content2' size ='200px' min='30%'
max='60%'></e-pane>
            <e-pane :content='content3' size ='200px'></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
var contentVue1 = Vue.component("contentTemp1", {
    template: `<div class="content">
                    <h3 class="h3">Grid </h3>
                    The ASP.NET DataGrid component, or DataTable is a
feature-rich component used to display data in a tabular format.
```

```
                </div>`,
    data() {
        return {
             data: {}
        };
    }
});
var contentVue2 = Vue.component("contentTemp2", {
    template: `<div class="content">
                    <h3 class="h3">Schedule </h3>
                    The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue3 = Vue.component("contentTemp3", {
    template: `<div class="content">
                    <h3 class="h3">Chart </h3>
                    ASP.NET charts, a well-crafted easy-to-use charting
package, is used to add beautiful charts in web and mobile applications
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
    name: 'app',
    data () {
        return {
        content1: function (){
            return { template: contentVue1 }
        },
        content2: function (){
            return { template: contentVue2 }
        },
        content3: function (){
            return { template: contentVue3 }
        },
        resize: false
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    margin: 65px auto;
}
.content {
    padding: 10px;
```

```
}
.e-splitter {
    margin: 0 auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/resize-min-max-cs8" %}

Refresh content on resizing

While resizing the panes, you can refresh the pane contents by using either resizeStart, resizing or resizeStop events.

Customize the resize grip and cursor

You can customize the resize gripper icon and cursor in css level.

APP.VUE

```
<template>
<div id="app" class="col-lg-12 control-section default-splitter">
    <ejs-splitter id='splitter' ref='splitterObj'  width='600px'
height='200px'>
        <e-panes>
            <e-pane :content='content1' size ='200px' min='20%'
max='40%'></e-pane>
            <e-pane :content='content2' size ='200px' min='20%'
max='60%'></e-pane>
            <e-pane :content='content3' size = '200px'></e-pane>
        </e-panes>
    </ejs-splitter>
</div>
</template>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
Vue.use(SplitterPlugin);
var contentVue1 = Vue.component("contentTemp1", {
    template: `<div class="content">
                    <h3 class="h3">PARIS </h3>
                    Paris, the city of lights and love - this short guide is
full of ideas for how to make the most of the romanticism...
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
var contentVue2 = Vue.component("contentTemp2", {
    template: `<div class="content">
                    <h3 class="h3">CAMEMBERT </h3>
                    The village in the Orne département of Normandy where
the famous French cheese is originated from.
                </div>`,
    data() {
        return {
            data: {}
```

```
        };
    }
});
var contentVue3 = Vue.component("contentTemp3", {
    template: `<div class="content">
                    <h3 class="h3">GRENOBLE </h3>
                    The capital city of the French Alps and a major
scientific center surrounded by many ski resorts, host of the Winter
Olympics in 1968.
                </div>`,
    data() {
        return {
            data: {}
        };
    }
});
export default {
    name: 'app',
    data () {
        return {
        content1: function (){
            return { template: contentVue1 }
        },
        content2: function (){
            return { template: contentVue2 }
        },
        content3: function (){
            return { template: contentVue3 }
        }
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-layouts/styles/material.css";
#app {
    margin: 65px auto;
}
.content {
    padding: 10px;
}
.e-splitter {
    margin: 0 auto;
}
.default-splitter .e-splitter .e-split-bar .e-resize-handler:before {
  content: "\e934";
  font-family: 'e-icons';
  font-size: 11px;
  transform: rotate(90deg);
}
.e-splitter .e-split-bar.e-split-bar-horizontal.e-resizable-split-bar,
.e-splitter .e-split-bar.e-split-bar-horizontal.e-resizable-split-bar::after
{
  cursor: e-resize;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/splitter/resize-min-max-cs9" %}

See Also

- Collapsible panes

Different layouts in Vue Splitter component

By using splitter component, you can create the different layouts with multiple and nested panes.

Code editor style layout

Step 1:

Create the element with two child to render the outer splitter and render the first pane of vertical splitter as a horizontal splitter.

```
<template>
<ejs-splitter id='outerSplitter' ref="splitterObj" :created='onCreate' orientation='Vertical' class='splitterContent' width='100%' height='400px'>
<e-panes>
<e-pane size="53%" min ="30%" ></e-pane>
<e-pane :content='pane4Content'></e-pane>
</e-panes>
</ejs-splitter>
</template>
<style>
.code-editor #code-text {
margin-left: 5px;
}
.code-editor .code-preview {
margin-top: 15px;
font-size: 12px;
}
.code-editor #target {
margin: 20px auto;
max-width: 600px;
}
.code-editor.control-section{
```

Copyright © 2001 -2024 Syncfusion Inc.

```
min-height: 370px;
margin-bottom: 15px;
margin-top: 10px;
}
.code-editor .h3 {
font-size: 14px;
margin: 4px;
}
.code-editor .content {
padding: 12px;
}
.code-editor .splitter-image {
margin: 0 auto;
display: flex;
height: 115px;
margin-top: 10px;
}
</style>
`
`javascript
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
import { Splitter } from '@syncfusion/ej2-layouts';
Vue.use(SplitterPlugin);
export default Vue.extend({
data: function() {
return {
pane4Content: function () {
return { template : pane4Content }
},
pane1Content: `<div><div class="content"><h3 class="h3">HTML</h3><div class="code-
preview"><<span>!DOCTYPE html></span><div><<span>html></span></div>
```

```
<div><<span>body></span></div><<span>div</span> id="custom-image"><div style="margin-left: 5px"><<span>img</span> src="src/albert.png"></div>

<div><<span>div</span>></div><div><<span>/body></span></div><div><<span>/html></span></div></div></div></div>`,

pane2Content: `<div><div class="content"><h3 class="h3">CSS</h3><div class="code-preview"><span>img {</span><div id="code-text">margin:<span>0 auto;</span></div>

<div id="code-text">display:<span>flex;</span></div><div id="code-text">height:<span>70px;</span></div><span>}</span></div></div></div>`,

pane3Content: `<div><div class="content"><h3 class="h3">JavaScript</h3><div class="code-preview"><span>var </span>

image = document.getElementById("custom-image");<div>image.addEventListener("click", function() {</div>

<div style="padding-left: 20px;">// Code block for click action</div><span>
}</span></div></div></div>`,

pane4Content: `

<div>

<div class="content">

<h3 class="h3">Preview of sample</h3>

<div class="splitter-image">

<img class="img1" src="https://ej2.syncfusion.com/demos/src/listview/images/albert.png" style="width: 20%;margin: 0 auto;">

</div>

</div>

</div>

`

}

},

methods: {

onCreate: function () {

document.getElementById('outerSplitter').querySelector('.e-pane-vertical').setAttribute('id', 'Innersplitter');

this.splitterObj1 = new Splitter({

height: '220px',

paneSettings: [

{ size: '29%', min: '23%', content: this.pane1Content },

{ size: '20%', min: '15%', content: this.pane2Content },
```

```
{ size: '35%', min: '35%', content: this.pane3Content }
],
width: '100%'
});
this.splitterObj1.appendTo('#Innersplitter');
}
}
});
</script>
```

Once the above configurations has been completed, you will get the output like this

Outlook style layout

Step 1:

Create the element with three panes and place the elements within the pane to render `treeview`, `listview` and `RTE`.

```
<template>
<ejs-splitter id='splitter' ref="splitterObj" width='100%' height='498px'>
<e-panes>
<e-pane size="28%" min="27%" :content='pane1Content'></e-pane>
<e-pane size="33%" min ="23%" :content='pane2Content'></e-pane>
<e-pane size="37%" min ="30%" :content='pane3Content'></e-pane>
</e-panes>
</ejs-splitter>
</template>
<style>
.outlook-style #target {
margin: 20px auto;
max-width: 820px;
}
.outlook-style #discard {
margin-left: 7px;
}
.outlook-style table {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
width: 100%;
border: none;
}
.outlook-style td {
padding: 2px;
}
.outlook-style.control-section {
min-height: 370px;
}
.outlook-style .e-treeview .e-list-text {
width: 100%;
}
.outlook-style #groupedList.e-listview .e-list-group-item {
height: 0;
}
.outlook-style #splitter1 .settings.e-list-wrapper.e-list-multi-line.e-list-avatar {
padding: 15px;
}
.outlook-style #buttonSection {
padding: 7px;
}
.outlook-style #createpostholder {
padding-left: 3px;
padding-right: 4px;
}
</style>
<script>
import Vue from "vue";
import { SplitterPlugin } from '@syncfusion/ej2-vue-layouts';
import pane1Content from "./outlook-pane1-content.vue";
import pane2Content from "./outlook-pane2-content.vue";
import pane3Content from "./outlook-pane3-content.vue";
Vue.use(SplitterPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default Vue.extend({
data: function() {
return {
pane1Content: function () {
return { template : pane1Content }
},
pane2Content: function () {
return { template : pane2Content }
},
pane3Content: function () {
return { template : pane3Content }
}
}
}
});
</script>
```

Step 2 :

Create the separate template files to render the inner components.

Below template defined for treeview

```
<template>
<div>
<div class="tree-template-control-wrapper">
<ejs-treeview id="template" :fields="fields" cssClass="custom" :nodeTemplate="treeTemplate"></ejs-treeview>
</div>
</div>
</template>
<style>
.tree-template-control-wrapper .e-treeview .e-list-text {
width: 100%;
}
```

```
</style>
<script>
import Vue from "vue";
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
var mailBox = [
{ id: 1, name: 'Favorites', hasChild: true},
{ id: 2, pid: 1, name: 'Sales Reports', count: '4' },
{ id: 3, pid: 1, name: 'Sent Items'},
{ id: 4, pid: 1, name: 'Marketing Reports ', count: '6'},
{ id: 5, name: 'Andrew Fuller', hasChild: true, expanded: true },
{ id: 6, pid: 5, name: 'Inbox',  selected: true , count: '20'},
{ id: 7, pid: 5,  name: 'Drafts', count: '5'},
{ id: 8, pid: 5,  name: 'Deleted Items'},
{ id: 9, pid: 5, name: 'Sent Items'},
{ id: 10, pid: 5, name: 'Sales Reports' , count: '4'},
{ id: 11, pid: 5, name: 'Marketing Reports', count: '6' },
{ id: 12, pid: 5, name: 'Outbox' },
{ id: 13, pid: 5, name: 'Junk Email'},
{ id: 14, pid: 5, name: 'RSS Feed'},
{ id: 15, pid: 5, name: 'Trash' }
];
var treeTemplate = Vue.component("demo", {
template: '<div><div class="treeviewdiv"><div style="float:left"><span
class="treeName">{{data.name}}</span></div>' +
'<div v-if="data.count" style="margin-right: 5px; float:right"><span class="treeCount e-badge e-badge-
primary">{{data.count}}</span></div></div></div>',
data() {
return {
data: {}
};
}
});
export default Vue.extend ({
```

Copyright © 2001 -2024 Syncfusion Inc.

```
data: function() {
return {
fields: { dataSource: mailBox, id: 'id', parentID: 'pid', text: 'name', hasChildren: 'hasChild' },
treeTemplate: function(e) {
return {
template: treeTemplate
};
},
};
}
});
</script>
`
Below template defined for listview
`
<template>
<div>
<ejs-listview id='listview' :dataSource='outlookdata' :cssClass='cssClass' :template='datatemplate'></ejs-listview>
</div>
</template>
<script>
import Vue from "vue";
import { ListViewPlugin } from "@syncfusion/ej2-vue-lists";
import { enableRipple } from "@syncfusion/ej2-base";
import datatempVue from "./data-vue-template.vue";
enableRipple(false);
Vue.use(ListViewPlugin);
export default Vue.extend({
data: function() {
return {
cssClass: "e-list-template",
outlookdata: [
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ Name: 'Selma Tally', content1: 'Apology marketing email', content2:'Hello Ananya Singleton' },
{ Name: 'Illa Russo', content1: 'Annual conference', content2: 'Hi jeani Moresa' },
{ Name: 'Camden Macmellon', content1: 'Reference request- Camden hester', content2: 'Hello Kerry Best,' },
{ Name: 'Garth Owen', content1: 'Application for job Title', content2: 'Hello Illa Russo' },
{ Name: 'Ursula Patterson', content1: 'Programmaer Position Applicant', content2: 'Hello Kerry best,' }
],
datatemplate: function() {
return {
template: datatempVue
};
}
};
}
});
</script>
`
Below template defined for RTE
`
<template>
<div>
<div style="width: 100%; padding: 15px;">
```

To...	
Cc...	
Subject	

```
</div>
<div class="forum">
<div id="createpostholder">
<ejs-richtexteditor ref="rteInstance" height='262px'></ejs-richtexteditor>
<div id="buttonSection">
<ejs-button :isPrimary="isPrimary" id="send" >Send</ejs-button>
<ejs-button id="discard" >Discard</ejs-button>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</div>
</div>
</div>
</template>
<script>
import Vue from "vue";
import { RichTextEditorPlugin, Link, Image, HtmlEditor, Toolbar } from "@syncfusion/ej2-vue-richtexteditor";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { TextBoxPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(TextBoxPlugin);
Vue.use(RichTextEditorPlugin);
Vue.use(ButtonPlugin);
export default Vue.extend({
data: function() {
return {
isPrimary: true
};
},
mounted: function() {
this.$nextTick(function () {
this.$refs.rteInstance.$el.ej2_instances[0].refresh();
})
},
provide:{
richtexteditor:[Link, Image, HtmlEditor, Toolbar]
}
});
</script>
`
```

Once the above configurations has been completed, you will get the output like this.

Copyright © 2001 -2024 Syncfusion Inc.

See Also

- Multiple panes in Splitter

Style in Vue Splitter component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the split bar

Use the following CSS to customize the split bar properties.

Horizontal split bar

`css

/ default split bar color /

.e-splitter .e-split-bar.e-split-bar-horizontal{

background: blue;

}

/ split bar color in hover and active state /

.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover,

.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active {

background: green;

}

`

Vertical split bar

`css

/ default split bar color /

.e-splitter .e-split-bar.e-split-bar-vertical {

background: blue;

}

/ split bar color in hover and active state /

.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover,

.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active {

background: green;

}

`

Customizing the split bar resize handle

Use the following CSS to customize the split bar resize handle.

Copyright © 2001 -2024 Syncfusion Inc.

Horizontal split bar resize handle

```css
/ default split bar resize handle color /
.e-splitter .e-split-bar.e-split-bar-horizontal .e-resize-handler {
color: rgba(20, 27, 233, 0.54);
}
/ default split bar resize handle color in hover and active state /
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-resize-handler,
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-resize-handler {
color: green;
}
```

Vertical split bar resize handle

```css
/ default split bar resize handle color /
.e-splitter .e-split-bar.e-split-bar-vertical .e-resize-handler {
color: rgba(20, 27, 233, 0.54);
}
/ default split bar resize handle color in hover and active state /
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-resize-handler,
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-resize-handler {
color: green;
}
```

Customizing the split bar arrows

Use the following CSS to customize the split bar arrows.

Horizontal split bar resize arrows

```css
/ split bar arrows /
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-left::before, .e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-left::before, .e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-left::after, .e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-left::after, .e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right::before, .e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-right::before, .e-
```

Copyright © 2001 -2024 Syncfusion Inc.

```
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right::after, .e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-right::after {

background-color: green;

}

/ split bar arrows - circular border /

.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-left, .e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right {

border-color: rgba(33, 227, 22, 0.5);

}
```

Vertical split bar resize arrows

```
`css

/ split bar arrows /

.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-up::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-up::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-down::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-down::after {

background-color: green;

}

/ split bar arrows - circular border /

.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down {

border-color: rgba(33, 227, 22, 0.5);

}
`
```

To hide the resize handle in Splitter

Use the following CSS to hide the resize handler in the split bar

Hide the horizontal split bar resize arrow

```
`css

.e-splitter .e-split-bar.e-split-bar-horizontal .e-resize-handler {

display: none;

}
`
```

Copyright © 2001 -2024 Syncfusion Inc.

Hide the vertical split bar resize arrow

`css

.e-splitter .e-split-bar.e-split-bar-vertical .e-resize-handler {

display: none;

}

`

Globalization in Vue Splitter component

RTL

Specifies the direction of the Splitter component using the enableRtl property. For writing systems that require it like Arabic, Hebrew, etc., the direction can be switched to right-to-left.

The following code shows how to enable RTL behavior.

APP.VUE

```
 html
<template>
    <div id="app" class="col-lg-12 control-section default-splitter">
        <ejs-splitter id='splitter' height='200px' enableRtl='true'>
            <e-panes>
                <e-pane content='Left Pane' size='200px'></e-pane>
                <e-pane content='Middle Pane' size='200px'></e-pane>
                <e-pane content='Right Pane' size='200px'></e-pane>
            </e-panes>
        </ejs-splitter>
    </div>
</template>
<script>
    import Vue from "vue";
    import { SplitterPlugin} from '@syncfusion/ej2-vue-layouts';
    Vue.use(SplitterPlugin);
    export default {
        name: 'app',
        data() {
            return {}
        }
    }
</script>
<style>
.e-pane {
    align-items: center;
    padding: 9px;
    display: flex;
}
    @import "../node_modules/@syncfusion/ej2-vue-
layouts/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/splitter/rtl-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

See Also

- Different Layouts

Accessibility in Vue Splitter component

The Splitter component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Splitter component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | |

| Section 508 Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| Accessibility Checker Validation | |

| Axe-core Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

 - All features of the component meet the requirement.

 - Some features of the component do not meet the requirement.

Copyright © 2001 -2024 Syncfusion Inc.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

Keyboard interaction

You can use the following key shortcuts to access the splitter without interruptions:

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Tab | Helps in focusing the splitter on the page and switching between the consecutive splitter bars. |

| Shift + Tab | Helps in focusing the previous splitter bar element on the splitter. |

| Right arrow | Helps in moving the active horizontal orientated splitter bar to its Right side. |

| Left arrow | Helps in moving the active horizontal orientated splitter bar to its Left side. |

| Up arrow | Helps in moving the active vertical orientated splitter bar to its Up side. |

| Down arrow | Helps in moving the active vertical orientated splitter bar to its Down side. |

| Enter | Helps to toggle between expand and collapse actions of the splitter bar when it is active. |

Ensuring accessibility

The Splitter component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Splitter component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Splitter component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/splitter.html" %}

See also

- Accessibility in Syncfusion Vue components

Spreadsheet

Overview of the Vue Spreadsheet component

The Vue Spreadsheet is an user interactive component to organize and analyze data in tabular format with configuration options for customization. It will load data by importing an Excel/CSV file or from local and remote data sources such as JSON, RESTful services, OData services, and more. The populated data can be exported as Excel with xlsx, xls, CSV and PDF formats.

Key features

- Data sources: Bind the Spreadsheet component with an array of objects or data from a web service using DataManager.
- Virtualization: Provides the option to load large amount of data without performance degradation.
- Selection: Provides the option to select a cell or range of cells.
- Editing: Provides the option to dynamically edit a cell.

Copyright © 2001 -2024 Syncfusion Inc.

- Formulas: Provides built-in calculation library with pre-defined formulas and named range support.
- Clipboard: Provides the option to perform clipboard operations.
- Cell formatting: Provides the option to customize the appearance of cells.
- Number formatting: Provides the option to format the cell value.
- Open: Provides the option to open Excel and CSV files in Spreadsheet.
- Save: Provides the option to save Spreadsheet data as Excel, CSV, and PDF documents.
- Sorting: Helps you to arrange the data to particular order in a selected range of cells.
- Filtering: Helps you to view specific rows in the Spreadsheet by hiding the other rows.
- Undo Redo: Provides the option to perform undo redo operations in Spreadsheet.
- Collaborative editing: Provides the option for real time changes across multiple users in the Spreadsheet.
- Hyperlink: Provides the option to navigate to web link or cell reference within the sheet or to other sheet in Spreadsheet.
- Resize: Allows you to change the row height and column width. Auto fit the rows and columns based on its content.
- Wrap text: Provides the option to display the large content as multiple lines in a single cell.
- Data validation: Provides the option to validate edited values based on data validation rules defined for a cell or range of cells.
- Find and replace: Provides the option to find the data and replace it across all sheets in Spreadsheet.
- Protect sheet: Provides the option to restrict user actions like cell editing, row and column insertion, deletion, and resizing.
- Borders: Provides the option to customize cell gridlines such as color and its style for enhanced UI.
- Show/hide: Provides the option to show/hide rows, columns and sheets.
- Insert/delete: Provides the option to insert/delete rows, columns and sheets.
- Merge cells: Provides the option to combine two or more cells located in the same row or column into a single cell.
- Conditional formatting: Provides the option to format a cell or range of cells based on conditions applied.
- Autofill: Provides the option to fill or copy a series or pattern of values and formats into adjacent cells in any direction.
- Clear: Provides the option to clear the content, formats, and hyperlinks applied to a cell or range of cells in a Spreadsheet.
- Aggregates: Provides the option to check the sum, average, count, and more for the selected cells or range in the sheet.
- Picture: Allows you to view, insert, and modify a picture in a Spreadsheet with customizing options.
- Chart: Transforms your Spreadsheet data to an intuitive overview for better understanding and to make smart business decisions.
- Freeze panes: Allows you to keep the specified rows and columns always visible at the top and left side of the sheet while scrolling through the sheet.
- Password protection: Allows you to protect the workbook with a password.
- Multi-line editing: Allows you to insert a line break between paragraphs of the text within a cell in a Spreadsheet.
- Calculate range selection: Helps you to select a range or multiple ranges when editing a formula in a cell.

Copyright © 2001 -2024 Syncfusion Inc.

- Right-to-left (RTL): Aligns content in the Spreadsheet component from right to left.
- Templates: Templates can be used to create custom user experiences in the Spreadsheet.
- Globalization: Personalize the Spreadsheet component with different languages, as well as culture-specific number, date, and time formatting.
- Accessibility: Provides with built-in accessibility support which helps to access all the Spreadsheet component features through the keyboard, screen readers, or other assistive technology devices.

Getting Started with the Vue Spreadsheet Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Spreadsheet component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies are required to use the Spreadsheet control in your application:

```
|-- @syncfusion/ej2-vue-spreadsheet
|-- @syncfusion/ej2-vue-base
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-grids
```

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

```bash
npm install -g @vue/cli
vue create quickstart
cd quickstart
npm run serve
```

or

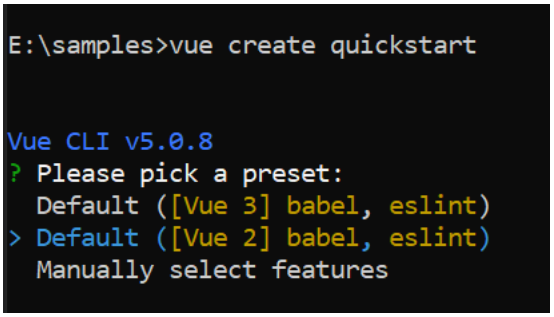
```bash
yarn global add @vue/cli
vue create quickstart
cd quickstart
```

Copyright © 2001 -2024 Syncfusion Inc.

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Spreadsheet component as an example. Install the @syncfusion/ej2-vue-spreadsheet package by running the following command:

`bash

npm install @syncfusion/ej2-vue-spreadsheet --save

`

or

`bash

yarn add @syncfusion/ej2-vue-spreadsheet

`

The **--save** will instruct NPM to include the spreadsheet package inside of the **dependencies** section of the **package.json**.

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Spreadsheet component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Spreadsheet component using Composition API or Options API:

1\. First, import and register the Spreadsheet component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SpreadsheetComponent as EjsSpreadsheet } from "@syncfusion/ej2-vue-
spreadsheet";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SpreadsheetComponent } from "@syncfusion/ej2-vue-spreadsheet";
export default {
components: {
'ejs-spreadsheet': SpreadsheetComponent
}
}
</script>
```

2\. In the template section define the Spreadsheet component

~/SRC/APP.VUE

```
<template>
<ejs-spreadsheet></ejs-spreadsheet>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
   <ejs-spreadsheet></ejs-spreadsheet>
</template>
<script setup>
import { SpreadsheetComponent as EjsSpreadsheet } from "@syncfusion/ej2-vue-
spreadsheet";
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
   <ejs-spreadsheet></ejs-spreadsheet>
</template>
<script>
import { SpreadsheetComponent } from "@syncfusion/ej2-vue-spreadsheet";
export default {
   components: {
      'ejs-spreadsheet': SpreadsheetComponent
   }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/spreadsheet/getting-started-cs1" %}

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example that shows you how to present and manipulate data.

See Also

- Data Binding
- Open and Save

Getting Started with the Vue Spreadsheet Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Spreadsheet component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

Copyright © 2001 -2024 Syncfusion Inc.

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

Copyright © 2001 -2024 Syncfusion Inc.

This article uses the Vue Spreadsheet component as an example. To use the Vue Spreadsheet component in the project, the @syncfusion/ej2-vue-spreadsheet package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-spreadsheet --save

`

or

`bash

yarn add @syncfusion/ej2-vue-spreadsheet

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Spreadsheet component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue Spreadsheet component using Composition API or Options API:

1.First, import and register the Spreadsheet component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { SpreadsheetComponent as EjsSpreadsheet, RangesDirective as ERanges,
RangeDirective as ERange, SheetsDirective as ESheets, SheetDirective as
ESheet} from "@syncfusion/ej2-vue-spreadsheet";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SpreadsheetComponent, RangesDirective, RangeDirective,
SheetsDirective, SheetDirective } from "@syncfusion/ej2-vue-spreadsheet";
export default {
name: "App",
// Declaring component and its directives
components: {
"ejs-spreadsheet": SpreadsheetComponent,
"e-sheets": SheetsDirective,
"e-sheet": SheetDirective,
"e-ranges": RangesDirective,
"e-range": RangeDirective,
}
}
</script>
```

2.In the template section, define the Spreadsheet component with sheets directives. Sheet directives are used to define the sheet definition for the Spreadsheet component.

~/SRC/APP.VUE

```
<template>
<ejs-spreadsheet>
<e-sheets>
<e-sheet>
<e-ranges>
<e-range :dataSource="data"></e-range>
</e-ranges>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>
</template>
```

3.Declare the values for the dataSource property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const data = [
{
OrderID: 10248,
Name: "VINET",
Country: "France",
},
{
OrderID: 10249,
Name: "TOMSP",
Country: "Germany",
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
];
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
data:[
{
OrderID: 10248,
Name: "VINET",
Country: "France",
},
{
OrderID: 10249,
Name: "TOMSP",
Country: "Germany",
}
],
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-spreadsheet>
<e-sheets>
<e-sheet>
<e-ranges>
<e-range :dataSource="data"></e-range>
</e-ranges>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>
</template>
<script setup>
import { SpreadsheetComponent as EjsSpreadsheet, RangesDirective as ERanges,
RangeDirective as ERange, SheetsDirective as ESheets, SheetDirective as
ESheet} from "@syncfusion/ej2-vue-spreadsheet";
const data = [
{
OrderID: 10248,
Name: "VINET",
Country: "France",
},
{
OrderID: 10249,
Name: "TOMSP",
Country: "Germany",
}
];
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-spreadsheet>
<e-sheets>
<e-sheet>
<e-ranges>
<e-range :dataSource="data"></e-range>
</e-ranges>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>
</template>
<script>
import { SpreadsheetComponent, RangesDirective, RangeDirective,
SheetsDirective, SheetDirective } from "@syncfusion/ej2-vue-spreadsheet";
export default {
name: "App",
// Declaring component and its directives
components: {
"ejs-spreadsheet": SpreadsheetComponent,
"e-sheets": SheetsDirective,
"e-sheet": SheetDirective,
"e-ranges": RangesDirective,
"e-range": RangeDirective,
},
// Bound properties declarations
data() {
return {
data:[
{
OrderID: 10248,
Name: "VINET",
Country: "France",
},
{
OrderID: 10249,
Name: "TOMSP",
Country: "Germany",
}
],
};
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

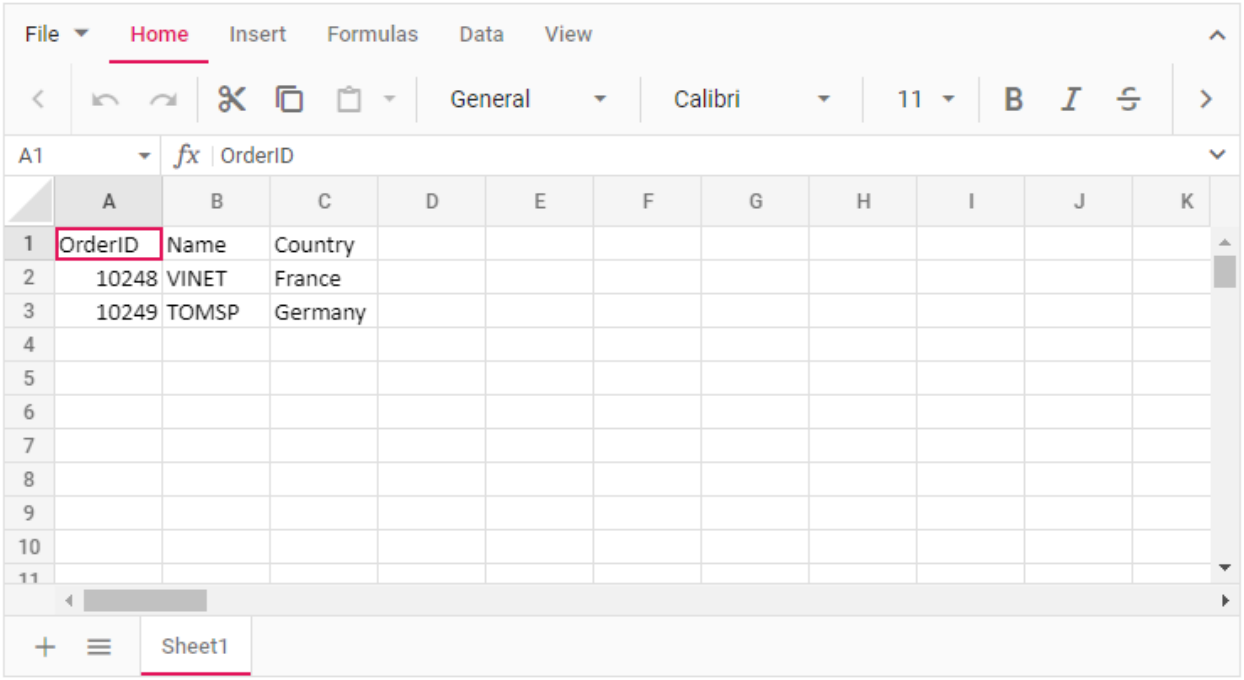
or

`bash

yarn run dev

`

The output will appear as follows:



For Migrating from Vue 2 to Vue 3 refer the Migration documentation

Data binding in Vue Spreadsheet component

The Spreadsheet uses [DataManager], which supports both RESTful JSON data services and local JavaScript object array binding to a range. The dataSource property can be assigned either with the instance of [DataManager] or JavaScript object array collection.

To bind data to a cell, use cell data binding support.

Local data

To bind local data to the Spreadsheet, you can assign a JavaScript object array to the dataSource property.

Refer to the following code example for local data binding.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet">
   <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: defaultData,
    }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/local-data-binding-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

The local data source can also be provided as an instance of the [DataManager]. By default, [DataManager] uses [JsonAdaptor] for local data-binding.

Remote data

To bind remote data to the Spreadsheet control, assign service data as an instance of [DataManager] to the dataSource property. To interact with remote data source, provide the service endpoint url.

Refer to the following code example for remote data binding.

APP.VUE

```
<template>
    <ejs-spreadsheet>
            <e-sheets>
                <e-sheet name="Shipment Details" :columns="columns">
                    <e-ranges>
                        <e-range :dataSource="dataSource" :query="query"
:showFieldAsHeader="false" startCell="A2"></e-range>
                    </e-ranges>
                    <e-rows>
                        <e-row>
                            <e-cells>
                                <e-cell value="Order ID"></e-cell>
                                <e-cell value="Customer Name"></e-cell>
                                <e-cell value="Freight"></e-cell>
                                <e-cell value="Ship Name"></e-cell>
                                <e-cell value="Ship City"></e-cell>
                                <e-cell value="Ship Country"></e-cell>
                            </e-cells>
                        </e-row>
                    </e-rows>
                </e-sheet>
            </e-sheets>
        </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { DataManager, Query } from "@syncfusion/ej2-data";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
     dataSource: new DataManager({
                    // Remote service url.
                    url:
"https://services.syncfusion.com/vue/production/api/Orders",
                    crossDomain: true
                }),
        query: new Query().select(['OrderID', 'CustomerID', 'Freight',
'ShipName', 'ShipCity', 'ShipCountry']).take(200),
        columns: [{ width: 100 }, { width: 130 }, { width: 100 }, { width:
220 }, { width: 150 }, { width: 180 }],
    }
  }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/remote-data-binding-cs1" %}

By default, DataManager uses **ODataAdaptor** for remote data-binding.

Binding with OData services

OData is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the DataManager. Refer to the following code example for remote Data binding using OData service.

APP.VUE

```
<template>
    <ejs-spreadsheet ref="spreadsheet">
            <e-sheets>
                <e-sheet name="Order Details" :columns="columns"
:created="created">
                    <e-ranges>
                        <e-range :dataSource="dataSource"></e-range>
                    </e-ranges>
                </e-sheet>
            </e-sheets>
        </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { DataManager, ODataAdaptor } from "@syncfusion/ej2-data";
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
     dataSource: new DataManager({
                    // Remote service url.
                    url:
"https://services.syncfusion.com/vue/production/api/Orders",
                    adaptor: new ODataAdaptor(),
                    crossDomain: true
                }),
        columns: [{ width: 80 }, { width: 80 }, { width: 80 }, { width: 80
}, { width: 80 }, { width: 80 }, { width: 280 }, { width: 180 }, { width: 80
}, { width: 180 }, { width: 180 }],
```

```
    }
  },
  methods: {
    created: function () {
        //Applies cell and number formatting to specified range of the
active sheet.
        this.$refs.spreadsheet.cellFormat({ fontWeight: 'bold', textAlign:
'center', verticalAlign: 'middle' },'A1:K1');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/remote-data-binding-cs2" %}

Web API

You can use WebApiAdaptor to bind spreadsheet with Web API created using OData endpoint.

APP.VUE

```
<template>
    <ejs-spreadsheet ref="spreadsheet">
            <e-sheets>
                <e-sheet name="Order Details" :columns="columns"
:created="created">
                    <e-ranges>
                        <e-range :dataSource="dataSource"></e-range>
                    </e-ranges>
                </e-sheet>
            </e-sheets>
        </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { DataManager, WebApiAdaptor } from "@syncfusion/ej2-data";
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
     dataSource: new DataManager({
                    // Remote service url.
```

```
                    url:
"https://services.syncfusion.com/vue/production/api/Orders",
                    adaptor: new WebApiAdaptor(),
                    crossDomain: true
                }),
        columns: [{ width: 80 }, { width: 80 }, { width: 80 }, { width: 80
}, { width: 80 }, { width: 80 }, { width: 280 }, { width: 180 }, { width: 80
}, { width: 180 }, { width: 180 }],
    }
  },
  methods: {
    created: function () {
        //Applies cell and number formatting to specified range of the
active sheet.
        this.$refs.spreadsheet.cellFormat({ fontWeight: 'bold', textAlign:
'center', verticalAlign: 'middle' },'A1:K1');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/remote-data-binding-cs3" %}

Cell data binding

The Spreadsheet control can bind the data to individual cell in a sheet . To achive this you can use the `value` property.

Refer to the following code example for cell data binding.

APP.VUE

```
<template>
     <ejs-spreadsheet >
            <e-sheets>
                <e-sheet name="Monthly Budget">
                    <e-rows>
                        <e-row>
                            <e-cells>
                                <e-cell value="Category"></e-cell>
                                <e-cell value="Planned cost"></e-cell>
                                <e-cell value="Actual cost"></e-cell>
                            </e-cells>
                        </e-row>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                        <e-row>
                            <e-cells>
                                <e-cell value="Food"></e-cell>
                                <e-cell value="$7000"></e-cell>
                                <e-cell value="$8120"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row>
                            <e-cells>
                                <e-cell value="Loan"></e-cell>
                                <e-cell value="$1500"></e-cell>
                                <e-cell value="$1500"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row>
                            <e-cells>
                                <e-cell value="Medical"></e-cell>
                                <e-cell value="$300"></e-cell>
                                <e-cell value="$0"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row>
                            <e-cells>
                                <e-cell value="Clothing"></e-cell>
                                <e-cell value="$400"></e-cell>
                                <e-cell value="$140"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row>
                            <e-cells>
                                <e-cell value="Education"></e-cell>
                                <e-cell value="$900"></e-cell>
                                <e-cell value="$750"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row>
                            <e-cells>
                                <e-cell value="Insurance"></e-cell>
                                <e-cell value="$30"></e-cell>
                                <e-cell value="$30"></e-cell>
                            </e-cells>
                        </e-row>
                    </e-rows>
                    <e-columns>
                        <e-column :width="width1"></e-column>
                        <e-column :width="width2"></e-column>
                        <e-column :width="width1"></e-column>
                    </e-columns>
                </e-sheet>
            </e-sheets>
        </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
   data: () => {
    return {
        width1: 110,
        width2: 115,
        width3: 100
    }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/cell-data-binding-cs1" %}

The cell data binding also supports formula, style, number format, and more.

Dynamic data binding and Datasource change event

You can dynamically change the datasource of the spreadsheet by changing the dataSource property of the range object of the sheet. The dataSourceChanged event handler will be triggered when editing, inserting, and deleting a row in the datasource range. This event will be triggered with a parameter named action which indicates the edit, add and delete actions for the respective ones.

The following table defines the arguments of the dataSourceChanged event.

| Property | Type | Description |

|-----|-----|-------|

| action | string | Indicates the type of action such as edit, add, and delete performed in the datasource range. |

| data | object[] | Modified data for edit action; New data for add action; Deleted data for delete action. |

| rangeIndex | number | Specifies the range index of the datasource. |

| sheetIndex | number | Specifies the sheet index of the datasource. |

For add action, the value for all the fields will be null in the data. In the case that you do not want the primary key field to be null which needs to be updated in the backend service, you can use edit action after updating the primary key field to update in the backend service.

For inserting a row at the end of the datasource range, you should insert a row below at the end of the range to trigger the dataSourceChanged event with action add.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div>
    <div>
      <ejs-button id="changeDataBtn" class="e-btn" v-
on:click.native="dataSourceBtnClick" style="margin-bottom: 10px" >Change
Datasource</ejs-button>
      <ejs-spreadsheet ref="spreadsheet"
:dataSourceChanged="dataSourceChanged">
        <e-sheets>
          <e-sheet name="Car Sales Report">
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
            <e-columns>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width3"></e-column>
            </e-columns>
          </e-sheet>
        </e-sheets>
      </ejs-spreadsheet>
    </div>
     <div>
                        <h4><b>Event Trace</b></h4>
                        <div id="evt" style="border: 1px solid
#dcdcdc;padding: 10px;">
                                <div style="height:173px;overflow: auto;min-
width: 250px;">
                                        <span id="EventLog" style="word-
break: normal;"></span>
                                </div>
                                <ejs-button id="clearBtn" class='e-
btn'>Clear</ejs-button>
                        </div>
                </div>
  </div>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { tradeData, defaultData } from './data.js';
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      width1: 110,
      width2: 115,
      width3: 100
      dataSource: tradeData
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
   methods: {
    dataSourceChanged: function (args) {
      this.appendElement("Data source changed with" + "<b> " +
args.action + "</b> action<hr>"
      );
    },
    dataSourceBtnClick: function () {
         this.$refs.spreadsheet.ej2Instances.sheets[0].ranges[0].dataSource
= defaultData;
    },
    clearBtnClick: function () {
      document.getElementById("EventLog").innerHTML = "";
    },
    appendElement: function (html) {
      var span = document.createElement("span");
      span.innerHTML = html;
      var log = document.getElementById("EventLog");
      log.insertBefore(span, log.firstChild);
    }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/dynamic-data-binding-cs1" %}

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Filtering
- Sorting
- Hyperlink
- Collaborative Editing

Open save in Vue Spreadsheet component

The native data format for Spreadsheet is JSON. When you open an excel file, it needs to be read and converted to client side Spreadsheet model. The converted client side Spreadsheet model is sent as

JSON which is used to render Spreadsheet. Similarly, when you save the Spreadsheet, the client Spreadsheet model is sent to the server as JSON for processing and saved as Excel file formats. Server configuration is used for this process.

Open

The Spreadsheet control opens an Excel document with its data, style, format, and more. To enable this feature, set allowOpen as true and assign service url to the openUrl property.

User Interface:

In user interface you can open an Excel document by clicking File > Open menu item in ribbon.

The following sample shows the Open option by using the openUrl property in the Spreadsheet control. You can also use the beforeOpen event to trigger before opening an Excel file.

APP.VUE

```
<template>
   <ejs-spreadsheet :openUrl="openUrl" :allowOpen="true"
:beforeOpen="beforeOpen"></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      openUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/open',
    }
  },
  methods: {
    beforeOpen: function (args) {
        // your code snippets here
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/open-cs1" %}

Please find the below table for the beforeOpen event arguments.

Copyright © 2001 -2024 Syncfusion Inc.

| **Parameter** | **Type** | **Description** |

| ----- | ----- | ----- |

| file | FileList or string or File | To get the file stream. FileList - contains length and item index.
 File - specifies the file lastModified and file name. |

| cancel | boolean | To prevent the open operation. |

| requestData | object | To provide the Form data. |

* Use Ctrl + O keyboard shortcut to open Excel documents.

* The default value of the allowOpen property is true. For demonstration purpose, we have showcased the allowOpen property in previous code snippet.

Open an external URL excel file while initial load

You can achieve to access the remote excel file by using the created event. In this event you can fetch the excel file and convert it to a blob. Convert this blob to a file and open this file by using Spreadsheet component open method.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :openUrl="openUrl" :allowOpen="true"
:created="created"></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      openUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/open',
    }
  },
  methods: {
    created: function () {
         fetch("https://cdn.syncfusion.com/scripts/spreadsheet/Sample.xlsx")
// fetch the remote url
                .then((response) => {
                    response.blob().then((fileBlob) => { // convert the
excel file to blob
                    var file = new File([fileBlob], "Sample.xlsx");
//convert the blob into file
                    this.$refs.spreadsheet.open({ file: file }); // open the
file into Spreadsheet
                    })
                })
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
```

```
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/open-cs2" %}

To add custom header during open

You can add your own custom header to the open action in the Spreadsheet. For processing the data, it has to be sent from server to client side and adding customer header can provide privacy to the data with the help of Authorization Token. Through the beforeOpen event, the custom header can be added to the request during open action.

APP.VUE

```
<template>
  <ejs-spreadsheet
    :openUrl="openUrl"
    :allowOpen="true"
    :beforeOpen="beforeOpen"
  ></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
Vue.use(SpreadsheetPlugin);
export default {
  data: () => {
    return {
      openUrl:

"https://services.syncfusion.com/vue/production/api/spreadsheet/open",
    };
  },
  methods: {
    beforeOpen: function (args) {
      args.requestData["headers"] = {
        Authorization: "YOUR TEXT",
      };
    },
  },
};
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-
vue-spreadsheet/styles/material.css";
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/custom-header-cs1" %}

Open excel file into a read-only mode

You can open excel file into a read-only mode by using the openComplete event. In this event, you must protect all the sheets and lock its used range cells by using protectSheet and lockCells methods

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :openUrl="openUrl" :allowOpen="true"
:openComplete="openComplete"></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin, getRangeAddress } from "@syncfusion/ej2-vue-
spreadsheet";
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      openUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/open',
    }
  },
  methods: {
    openComplete: function () {
        let spreadsheet = this.$refs.spreadsheet.ej2Instances;
        let sheets = spreadsheet.sheets;
        for (let index = 0; index < sheets.length; index++) {
            let name = spreadsheet.sheets[index].name;
            let protectSetting = {
                selectCells: true,
                formatCells: false,
            };
            //To protect the sheet using sheet name
            spreadsheet.protectSheet(name, protectSetting);
            let address = getRangeAddress([
                0,
                0,
                sheets[index].usedRange.rowIndex,
                sheets[index].usedRange.colIndex,
            ]);
            //To lock the used range cells
            spreadsheet.lockCells(name + '!' + address, true);
          }
        }
    }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/open-readonly-cs1" %}

External workbook confirmation dialog

When you open an excel file that contains external workbook references, you will see a confirmation dialog. This dialog allows you to either continue with the file opening or cancel the operation. This confirmation dialog will appear only if you set the AllowExternalWorkbook property value to **false** during the open request, as shown below. This prevents the spreadsheet from displaying inconsistent data.

`csharp

public IActionResult Open(IFormCollection openRequest)

{

OpenRequest open = new OpenRequest();

open.AllowExternalWorkbook = false;

open.File = openRequest.Files[0];

return Content(Workbook.Open(open));

}

`

This feature is only applicable when importing an Excel file and not when loading JSON data or binding cell data.

![External workbook confirmation dialog](./images/external-reference-dialog-alert%20.png)

Supported file formats

The following list of Excel file formats are supported in Spreadsheet:

- MS Excel (.xlsx)
- MS Excel 97-2003 (.xls)
- Comma Separated Values (.csv)
- Excel Macro-Enabled Workbook (.xlsm)
- Excel Binary Workbook(.xlsb)

Copyright © 2001 -2024 Syncfusion Inc.

Save

The Spreadsheet control saves its data, style, format, and more as Excel file document. To enable this feature, set allowSave as true and assign service url to the saveUrl property.

User Interface:

In user interface, you can save Spreadsheet data as Excel document by clicking File > Save As menu item in ribbon.

The following sample shows the Save option by using the saveUrl property in the Spreadsheet control. You can also use the beforeSave event to trigger before saving the Spreadsheet as an Excel file.

APP.VUE

```
<template>
   <ejs-spreadsheet :saveUrl="saveUrl" :allowSave="true"
:beforeSave="beforeSave">
   <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
            <e-columns>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width3"></e-column>
            </e-columns>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
      width1: 180,
      width2: 130,
      width3: 120,
      saveUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/save'
    }
  },
  methods: {
    beforeSave: function (args) {
        // your code snippets here
      }
    }
}
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/open-save-cs1" %}

Please find the below table for the beforeSave event arguments.

| **Parameter** | **Type** | **Description** |

| ----- | ----- | ----- |

| url | string | Specifies the save url. |

| fileName | string | Specifies the file name. |

| saveType | SaveType | Specifies the saveType like Xlsx, Xls, Csv and Pdf. |

| customParams | object | Passing the custom parameters from client to server while performing save operation. |

| isFullPost | boolean | It sends the form data from client to server, when set to true. It fetches the data from client to server and returns the data from server to client, when set to false. |

| needBlobData | boolean | You can get the blob data if set to true. |

| cancel | boolean | To prevent the save operations. |

* Use Ctrl + S keyboard shortcut to save the Spreadsheet data as Excel file.

* The default value of allowSave property is true. For demonstration purpose, we have showcased the allowSave property in previous code snippet.

* Demo purpose only, we have used the online web service url link.

To send and receive custom params from client to server

Passing the custom parameters from client to server by using beforeSave event.

APP.VUE

```
<template>
   <ejs-spreadsheet :saveUrl="saveUrl" :allowSave="true"
:beforeSave="beforeSave">
   <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
            <e-columns>
              <e-column :width="width1"></e-column>
```

```
                <e-column :width="width2"></e-column>
                <e-column :width="width2"></e-column>
                <e-column :width="width1"></e-column>
                <e-column :width="width2"></e-column>
                <e-column :width="width3"></e-column>
              </e-columns>
            </e-sheet>
          </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
      width1: 180,
      width2: 130,
      width3: 120,
      saveUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/save'
    }
  },
  methods: {
    beforeSave: function (args) {
        args.customParams = { customParams: 'you can pass custom params in
server side'}; // you can pass the custom params
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/open-save-cs2" %}

Server side code snippets:

`c#

public IActionResult Save(SaveSettings saveSettings, string customParams)

{

Console.WriteLine(customParams); // you can get the custom params in controller side

return Workbook.Save(saveSettings);

}

`

To add custom header during save

You can add your own custom header to the save action in the Spreadsheet. For processing the data, it has to be sent from client to server side and adding customer header can provide privacy to the data with the help of Authorization Token. Through the fileMenuItemSelect event, the custom header can be added to the request during save action.

APP.VUE

```
<template>
  <ejs-spreadsheet
    ref="spreadsheet"
    :saveUrl="saveUrl"
    :allowSave="true"
    :fileMenuItemSelect="fileMenuItemSelect"
  >
    <e-sheets>
      <e-sheet>
        <e-ranges>
          <e-range :dataSource="dataSource"></e-range>
        </e-ranges>
        <e-columns>
          <e-column :width="width1"></e-column>
          <e-column :width="width2"></e-column>
          <e-column :width="width2"></e-column>
          <e-column :width="width1"></e-column>
          <e-column :width="width2"></e-column>
          <e-column :width="width3"></e-column>
        </e-columns>
      </e-sheet> </e-sheets
  ></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { createElement } from "@syncfusion/ej2-base";
import { data } from "./data.js";
Vue.use(SpreadsheetPlugin);
export default {
  data: () => {
    return {
      dataSource: data,
      width1: 180,
      width2: 130,
      width3: 120,
      saveUrl:

"https://services.syncfusion.com/vue/production/api/spreadsheet/save",
    };
  },
  methods: {
```

```
    fileMenuItemSelect: function (args) {
      var spreadsheet = this.$refs.spreadsheet;
      if (args.item.text === "Microsoft Excel") {
        args.cancel = true;
        spreadsheet.saveAsJson().then((response) => {
          var formData = new FormData();
          formData.append(
            "JSONData",
            JSON.stringify(response.jsonObject.Workbook)
          );
          formData.append("fileName", "Sample");
          formData.append("saveType", "Xlsx");
          fetch(

"https://services.syncfusion.com/vue/production/api/spreadsheet/save",
            {
              method: "POST",
              headers: { Authorization: "YOUR TEXT" },
              body: formData,
            }
          ).then((response) => {
            response.blob().then((data) => {
              var anchor = createElement("a", {
                attrs: { download: "Sample.xlsx" },
              });
              var url = URL.createObjectURL(data);
              anchor.href = url;
              document.body.appendChild(anchor);
              anchor.click();
              URL.revokeObjectURL(url);
              document.body.removeChild(anchor);
            });
          });
        });
      }
    },
  },
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/custom-header-cs2" %}

Copyright © 2001 -2024 Syncfusion Inc.

To change the PDF orientation

By default, the PDF document is created in **Portrait** orientation. You can change the orientation of the PDF document by using the args.pdfLayoutSettings.orientation argument settings in the beforeSave event.

The possible values are:

- **Portrait** - Used to display content in a vertical layout.
- **Landscape** - Used to display content in a horizontal layout.

APP.VUE

```
<template>
   <ejs-spreadsheet :saveUrl="saveUrl" :allowSave="true"
:beforeSave="beforeSave">
   <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
            <e-columns>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width3"></e-column>
            </e-columns>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
      width1: 180,
      width2: 130,
      width3: 120,
      saveUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/save'
    }
  },
  methods: {
    beforeSave: function (args) {
        args.pdfLayoutSettings.orientation = 'Landscape'; // You can change
the orientation of the PDF document
      }
    }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/open-save-cs3" %}

Supported file formats

The following list of Excel file formats are supported in Spreadsheet:

- MS Excel (.xlsx)
- MS Excel 97-2003 (.xls)
- Comma Separated Values (.csv)
- Portable Document Format (.pdf)

Methods

To save the Spreadsheet document as an xlsx, xls, csv, or pdf file, by using save method should be called with the url, fileName and saveType as parameters. The following code example shows to save the spreadsheet file as an xlsx, xls, csv, or pdf in the button click event.

APP.VUE

```
<template>
  <div>
    <ejs-dropdownbutton :items='items' :select='itemSelect'>Save</ejs-
dropdownbutton>
    <ejs-spreadsheet ref="spreadsheet">
      <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
             <e-columns>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
            </e-columns>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
        <div>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin, getRangeIndexes } from "@syncfusion/ej2-vue-
spreadsheet";
import { addClass, removeClass } from '@syncfusion/ej2-base';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { DropDownButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
Vue.use(DropDownButtonPlugin);
export default {
   data: () => {
    return {
      dataSource: defaultData,
      width1: 130,
      width2: 96,
      items:[
        {
            text: "Save As xlsx"
        },
        {
            text: "Save As xls"
        },
        {
            text: "Save As csv"
        },
        {
            text: "Save As pdf"
        }
]
    }
  },
  methods: {
    itemSelect: function(args) {
      var spreadsheet = this.$refs.spreadsheet;
      if (args.item.text === 'Save As xlsx')
      spreadsheet.save({url:
'https://services.syncfusion.com/vue/production/api/spreadsheet/save',
fileName: "Sample", saveType: "Xlsx"});
    if (args.item.text === 'Save As xls')
      spreadsheet.save({url:
'https://services.syncfusion.com/vue/production/api/spreadsheet/save',
fileName: "Sample", saveType: "Xls"});
    if (args.item.text === 'Save As csv')
      spreadsheet.save({url:
'https://services.syncfusion.com/vue/production/api/spreadsheet/save',fileNa
me: "Sample", saveType: "Csv"});
    if (args.item.text === 'Save As pdf')
      spreadsheet.save({url:
'https://services.syncfusion.com/vue/production/api/spreadsheet/save',fileNa
me: "Sample", saveType: "Pdf"});
   }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
  @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/save-cs1" %}

Server Configuration

Import and export are processed in server-side using Spreadsheet server library. The following code snippets shows server configuration using WebAPI service,

`c#

[Route("api/[controller]")]

public class SpreadsheetController : Controller

{

//To open excel file

[AcceptVerbs("Post")]

[HttpPost]

[EnableCors("AllowAllOrigins")]

[Route("Open")]

public IActionResult Open(IFormCollection openRequest)

{

OpenRequest open = new OpenRequest();

open.File = openRequest.Files[0];

return Content(Workbook.Open(open));

}

//To save as excel file

[AcceptVerbs("Post")]

[HttpPost]

[EnableCors("AllowAllOrigins")]

[Route("Save")]

public IActionResult Save([FromForm]SaveSettings saveSettings)

{

return Workbook.Save(saveSettings);

}

}

`

Copyright © 2001 -2024 Syncfusion Inc.

Server Dependencies

Open and save helper functions are shipped in the Syncfusion.EJ2.Spreadsheet package, which is available in Essential Studio and nuget.org. Following list of dependencies required for Spreadsheet open and save operations.

- Syncfusion.EJ2
- Syncfusion.EJ2.Spreadsheet
- Syncfusion.Compression.Base
- Syncfusion.XlsIO.Base

And also refer this for more information.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Filtering
- Sorting
- Hyperlink

Worksheet in Vue Spreadsheet component

Worksheet is a collection of cells organized in the form of rows and columns that allows you to store, format, and manipulate the data.

Add sheet

You can dynamically add or insert a sheet by one of the following ways,

- Click the Add Sheet button in the sheet tab. This will add a new empty sheet next to current active sheet.
- Right-click on the sheet tab, and then select Insert option from the context menu to insert a new empty sheet before the current active sheet.
- Using insertSheet method, you can insert one or more sheets at your desired index.

The following code example shows the insert sheet operation in spreadsheet.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false" :showRibbon="false">
                <e-sheets>
                  <e-sheet name="Price Details">
                    <e-ranges>
                      <e-range :dataSource="dataSource"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=150></e-column>
                      <e-column :width=110></e-column>
                      <e-column :width=110></e-column>
```

```
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                      </e-columns>
                    </e-sheet>
                  </e-sheets>
                </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
    }
  },
  methods: {
  created: function () {
      var spreadsheet = this.$refs.spreadsheet;
       // Applies style formatting to the active sheet before inserting a
new sheet
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'D2:H11');
        // inserting a new sheet with data at 1st index
        // You can also insert empty sheets by specifying the start and end
sheet index instead of sheet model
        spreadsheet.insertSheet([{
            index: 1,
            name: 'Inserted Sheet',
            ranges: [{ dataSource: this.dataSource }],
            columns: [{ width: 150 }, { width: 110 }, { width: 110 }, {
width: 85 }, { width: 85 }, { width: 85 }, { width: 85 },
                { width: 85 }]
        }]);
        // Applies style formatting for the inserted sheet
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'Inserted Sheet!A1:H1');
       spreadsheet.cellFormat({ textAlign: 'center' }, 'Inserted
Sheet!D2:H11');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/insert-sheet-cs1" %}

Delete sheet

The Spreadsheet has support for removing an existing worksheet. You can dynamically delete the existing sheet by the following way,

- Right-click on the sheet tab, and then select Delete option from context menu.
- Using delete method to delete the sheets.

Rename sheet

You can dynamically rename an existing worksheet in the following way,

- Right-click on the sheet tab, and then select Rename option from the context menu.

Headers

By default, the row and column headers are visible in worksheets. You can dynamically show or hide worksheet headers by using one of the following ways,

- Switch to View tab, and then select Hide Headers option to hide both the row and column headers.
- Set showHeaders property in sheets as true or false to show or hide the headers at initial load. By default, the showHeaders property is enabled in each worksheet.

Gridlines

Gridlines act as a border like appearance of cells. They are used to distinguish cells on the worksheet. You can dynamically show or hide gridlines by using one of the following ways,

- Switch to View tab, and then select Hide Gridlines option to hide the gridlines in worksheet.
- Set showGridLines property in sheets as true or false to show or hide the gridlines at initial load. By default, the showGridLines property is enabled in each worksheet.

The following code example shows the headers and gridlines operation in spreadsheet.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false"
                :showSheetTabs="false">
                <e-sheets>
                  <!-- Hiding the headers and gridlines in 'Price Details'
sheet -->
                  <e-sheet :showGridLines="false" :showHeaders="false">
                    <e-ranges>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                        <e-range :dataSource="dataSource"></e-range>
                      </e-ranges>
                      <e-columns>
                        <e-column :width=150></e-column>
                        <e-column :width=110></e-column>
                        <e-column :width=110></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                      </e-columns>
                    </e-sheet>
                  </e-sheets>
                </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
    }
  },
  methods: {
  created: function () {
      var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'D2:H11');
        // The gridlines have been removed to set border for the range of
cells
        spreadsheet.setBorder({ border: '1px solid #e0e0e0' }, 'A1:H11');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/header-gridlines-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Sheet visibility

Hiding a worksheet can help prevent unauthorized or accidental changes to your file.

There are three visibility state as like Microsoft Excel,

| State | Description |

|-------|---------|

| `Visible` | You can see the worksheet once the component is loaded. |

| `Hidden` | This worksheet is not visible, but you can unhide by selecting the sheet from `List All Sheets` dropdown menu. |

| `VeryHidden` | This worksheet is not visible and cannot be unhidden. Changing the state property to `Visible` is the only way to view this sheet. |

The following code example shows the three types of sheet visibility state.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :created="created" :openUrl="openUrl"
                  :saveUrl="saveUrl" :showFormulaBar="false"
:showRibbon="false">
                  <e-sheets>
                    <!-- By default, state is set as 'visible'. We don’t  need
to said it in the sample. -->
                    <e-sheet name="Visible Sheet" state="Visible">
                      <e-ranges>
                        <e-range :dataSource="dataSource"></e-range>
                      </e-ranges>
                      <e-columns>
                        <e-column :width=150></e-column>
                        <e-column :width=110></e-column>
                        <e-column :width=110></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                      </e-columns>
                    </e-sheet>
                    <!-- Sets sheet state as 'VeryHidden'. It can't be
unhidden. -->
                    <e-sheet name="Very Hidden Sheet" state="VeryHidden">
                      <e-ranges>
                        <e-range :dataSource="dataSource"></e-range>
                      </e-ranges>
                      <e-columns>
                        <e-column :width=150></e-column>
                        <e-column :width=110></e-column>
                        <e-column :width=110></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
                        <e-column :width=85></e-column>
```

```
                    </e-columns>
                  </e-sheet>
                  <!-- Sets sheet state as 'Hidden'. It can be unhidden
dynamically. -->
                  <e-sheet name="Hidden Sheet" state="Hidden">
                    <e-ranges>
                      <e-range :dataSource="dataSource"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=150></e-column>
                      <e-column :width=110></e-column>
                      <e-column :width=110></e-column>
                      <e-column :width=85></e-column>
                      <e-column :width=85></e-column>
                      <e-column :width=85></e-column>
                      <e-column :width=85></e-column>
                      <e-column :width=85></e-column>
                    </e-columns>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
        openUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/open',
        saveUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/save',
    }
  },
  methods: {
  created: function () {
      var spreadsheet = this.$refs.spreadsheet;
      // Applies style formatting to active visible sheet.
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'D2:H11');
        // Applies style formatting to active hidden sheet.
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'Hidden Sheet!A1:H1');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'Hidden
Sheet!D2:H11');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
```

```
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/sheet-visiblity-cs1" %}

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Sheet protection
- Rows and columns
- Cell range
- Formatting

Cell range in Vue Spreadsheet component

A group of cells in a sheet is known as cell range.

Wrap text

Wrap text allows you to display large content as multiple lines in a single cell. By default, the wrap text support is enabled. Use the allowWrap property to enable or disable the wrap text support in spreadsheet.

Wrap text can be applied or removed to a cell or range of cells in the following ways,

- Using the `wrap` property in `cell`, you can enable or disable wrap text to a cell at initial load.
- Select or deselect wrap button from ribbon toolbar to apply or remove the wrap text to the selected range.
- Using the wrap method, you can apply or remove the wrap text once the component is loaded.

The following code example shows the wrap text functionality in spreadsheet.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false">
  <e-sheets>
    <e-sheet name="Movie List">
      <e-ranges>
        <e-range :dataSource="dataSource"></e-range>
      </e-ranges>
      <e-columns>
        <e-column index=1 :width="100"></e-column>
        <e-column :width="140"></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <e-column :width="90"></e-column>
        <e-column :width="150"></e-column>
        <e-column :width="120"></e-column>
        <e-column :width="90"></e-column>
        <e-column :width="180"></e-column>
      </e-columns>
      <e-rows>
        <e-row :height="30"></e-row>
        <e-row>
          <e-cells>
            <e-cell :index=7 wrap="true"></e-cell>
          </e-cells>
        </e-row>
        <e-row>
          <e-cells>
            <e-cell :index=7 wrap="true"></e-cell>
          </e-cells>
        </e-row>
        <e-row>
          <e-cells>
            <e-cell :index=7 wrap="true"></e-cell>
          </e-cells>
        </e-row>
        <e-row>
          <e-cells>
            <e-cell :index=7 wrap="true"></e-cell>
          </e-cells>
        </e-row>
        <e-row>
          <e-cells>
            <e-cell :index=7 wrap="true"></e-cell>
          </e-cells>
        </e-row>
      </e-rows>
    </e-sheet>
  </e-sheets>
</ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
    }
  },
  methods: {
  created: function () {
      var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
      spreadsheet.cellFormat({ verticalAlign: 'middle' }, 'A1:H5');
      spreadsheet.cellFormat({ textAlign: 'center' }, 'A2:B5');
      spreadsheet.cellFormat({ textAlign: 'center' }, 'D2:D5');
      // To wrap the cells from E2 to E5 range
      spreadsheet.wrap('E2:E5');
      // To unwrap the H3 cell
      spreadsheet.wrap('H3', false);
      }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/wrap-text-cs1" %}

Limitations of Wrap text

The following features have some limitations in wrap text:

- Sorting with wrap text applied data.
- Merge with wrap text.

Merge cells

Merge cells allows users to span two or more cells in the same row or column into a single cell. When cells with multiple values are merged, top-left most cell data will be the data for the merged cell. By default, the merge cells option is enabled. Use allowMerge property to enable or disable the merge cells option in spreadsheet.

You can merge the range of cells in the following ways,

- Set the `rowSpan` and `colSpan` property in `cell` to merge the number of cells at initial load.
- Select the range of cells and apply merge by selecting the desired option from ribbon toolbar.
- Use merge method to merge the range of cells, once the component is loaded.

The available merge options in spreadsheet are,

| Type | Action |

|-------|---------|

| Merge All | Combines all the cells in a range in to a single cell (default). |

| Merge Horizontally | Combines cells in a range as row-wise. |

| Merge Vertically | Combines cells in a range as column-wise. |

| UnMerge | Splits the merged cells into multiple cells. |

The following code example shows the merge cells operation in spreadsheet.

APP.VUE

```
<template>
```

```
    <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false">
  <e-sheets>
    <e-sheet name="Merge Cells">
      <e-ranges>
        <e-range :dataSource="dataSource"></e-range>
      </e-ranges>
      <e-columns>
        <e-column :width=90></e-column>
        <e-column :width=150></e-column>
        <e-column :width=100></e-column>
        <e-column :width=100></e-column>
        <e-column :width=100></e-column>
        <e-column :width=100></e-column>
        <e-column :width=100></e-column>
        <e-column :width=100></e-column>
        <e-column :width=100></e-column>
        <e-column :width=100></e-column>
        <e-column :width=120></e-column>
        <e-column :width=120></e-column>
        <e-column :width=120></e-column>
        <e-column :width=120></e-column>
        <e-column :width=120></e-column>
        <e-column :width=120></e-column>
        <e-column :width=120></e-column>
        <e-column :width=100></e-column>
        <e-column :width=100></e-column>
        <e-column :width=100></e-column>
        <e-column :width=100></e-column>
      </e-columns>
      <e-rows>
        <e-row :height=35></e-row>
        <e-row :height=35>
          <e-cells>
            <e-cell :index=1 :rowSpan=2></e-cell>
            <e-cell :colSpan=2></e-cell>
            <e-cell :index=6 :colSpan=3></e-cell>
            <e-cell :index=10 :rowSpan=2 :colSpan=3></e-cell>
            <e-cell :index=13 :colSpan=2></e-cell>
            <e-cell :index=17 :colSpan=2></e-cell>
          </e-cells>
        </e-row>
        <e-row :height=35>
          <e-cells>
            <e-cell :index=3 :colSpan=3></e-cell>
            <e-cell :index=6 :colSpan=4></e-cell>
            <e-cell :index=13 :colSpan=3></e-cell>
            <e-cell :index=17 :colSpan=2></e-cell>
          </e-cells>
        </e-row>
        <e-row :height=35>
          <e-cells>
            <e-cell :index=2 :colSpan=3></e-cell>
            <e-cell :index=5 :colSpan=2></e-cell>
            <e-cell :index=7 :colSpan=3></e-cell>
            <e-cell :index=15 :colSpan=2></e-cell>
            <e-cell :index=17 :colSpan=2></e-cell>
          </e-cells>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </e-row>
        <e-row :height=35>
          <e-cells>
            <e-cell :index=2 :colSpan=3></e-cell>
            <e-cell :index=6 :colSpan=4></e-cell>
            <e-cell :index=16 :colSpan=2></e-cell>
          </e-cells>
        </e-row>
        <e-row :height=35>
          <e-cells>
            <e-cell :index=2 :colSpan=4></e-cell>
            <e-cell :index=7 :colSpan=3></e-cell>
            <e-cell :index=15 :colSpan=2></e-cell>
            <e-cell :index=17 :colSpan=2></e-cell>
          </e-cells>
        </e-row>
      </e-rows>
    </e-sheet>
  </e-sheets>
</ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
    }
  },
  methods: {
  created: function () {
      var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:S1');
      spreadsheet.numberFormat('h:mm AM/PM', 'C1:S1');
      spreadsheet.cellFormat({ verticalAlign: 'middle' }, 'A1:S11');
        // Merging the `K4:M4` cells using method
        spreadsheet.merge('K4:M4');
      // Merging the 5th and 6th row cells across 11th, 12th and 13th column
      spreadsheet.merge('K5:M6', 'Vertically');
      // Merging the 18th and 19th column cells across 2nd, 3rd and 4th row
      spreadsheet.merge('N4:O6', 'Horizontally');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/merge-cells-cs1" %}

Limitations of Merge

The following features have some limitations in Merge:

- Merge with filter.
- Merge with wrap text.

Data Validation

Data Validation is used to restrict the user from entering the invalid data. You can use the allowDataValidation property to enable or disable data validation.

* The default value for `allowDataValidation` property is `true`.

Apply Validation

You can apply data validation to restrict the type of data or the values that users enter into a cell.

You can apply data validation by using one of the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Data Validation item.
- Use the addDataValidation() method programmatically.

Clear Validation

Clear validation feature is used to remove data validations from the specified ranges or the whole worksheet.

You can clear data validation rule by one of the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Clear Validation item.
- Use the removeDataValidation() method programmatically.

Highlight Invalid Data

Highlight invalid data feature is used to highlight the previously entered invalid values.

You can highlight an invalid data by using one of the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Highlight Invalid Data item.
- Use the addInvalidHighlight() method programmatically.

Clear Highlighted Invalid Data

Clear highlight feature is used to remove the highlight from invalid cells.

You can clear the highlighted invalid data by using the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Clear Highlight item.

Copyright © 2001 -2024 Syncfusion Inc.

- Use the removeInvalidHighlight() method programmatically.

APP.VUE

```
<template>
 <ejs-spreadsheet ref="spreadsheet" :created="created">
                <e-sheets>
                  <e-sheet name ="PriceDetails">
                    <e-columns>
                      <e-column :width=88></e-column>
                      <e-column :width=88></e-column>
                      <e-column :width=106></e-column>
                      <e-column :width=98></e-column>
                      <e-column :width=88></e-column>
                      <e-column :width=86></e-column>
                      <e-column :width=107></e-column>
                      <e-column :width=81></e-column>
                    </e-columns>
                    <e-rows>
                      <e-row>
                        <e-cells>
                          <e-cell value="Seller Name" style="style"></e-
cell>
                          <e-cell value="Customer Id" style="style"></e-
cell>
                          <e-cell value="Customer Name" style="style"></e-
cell>
                          <e-cell value="Product Name" style="style"></e-
cell>
                          <e-cell value="Product Price" style="style"></e-
cell>
                          <e-cell value="Sales Date" style="style"></e-cell>
                          <e-cell value="Billing Time" style="style"></e-
cell>
                          <e-cell value="Total Price" style="style"></e-
cell>
                        </e-cells>
                      </e-row>
                      <e-row>
                       <e-cells>
                          <e-cell value="John"></e-cell>
                          <e-cell value="1" validation="validation"></e-
cell>
                          <e-cell value="Nash"></e-cell>
                          <e-cell value="Digger"
validation="listValidation"></e-cell>
                          <e-cell value="50000"
validation="listValidation1"></e-cell>
                          <e-cell value="04/11/2019"></e-cell>
                          <e-cell value="11:34:32 AM"></e-cell>
                          <e-cell value="1,45,000.00"></e-cell>
                        </e-cells>
                      </e-row>
                      <e-row>
                       <e-cells>
                          <e-cell value="Mike"></e-cell>
```

```
                            <e-cell value="2" validation="validation"></e-
cell>
                            <e-cell value="Jim"></e-cell>
                            <e-cell value="Cherrypicker"
validation="validation1"></e-cell>
                            <e-cell value="45000"
validation="listValidation2"></e-cell>
                            <e-cell value="04/11/2019"></e-cell>
                            <e-cell value="10:15:00 AM"></e-cell>
                            <e-cell value="1,40,040.00"></e-cell>
                          </e-cells>
                        </e-row>
                        <e-row>
                         <e-cells>
                            <e-cell value="shane"></e-cell>
                            <e-cell value="3" validation="validation"></e-
cell>
                            <e-cell value="Sean"></e-cell>
                            <e-cell value="Kango"
validation="listValidation3"></e-cell>
                            <e-cell value="450"
validation="listValidation4"></e-cell>
                            <e-cell value="06/25/2019"></e-cell>
                            <e-cell value="01:30:11 PM"></e-cell>
                            <e-cell value="545.00"></e-cell>
                          </e-cells>
                        </e-row>
                        <e-row>
                         <e-cells>
                            <e-cell value="John"></e-cell>
                            <e-cell value="1" validation="validation"></e-
cell>
                            <e-cell value="Nash"></e-cell>
                            <e-cell value="JCB"
validation="listValidation5"></e-cell>
                            <e-cell value="90000"
validation="listValidation6"></e-cell>
                            <e-cell value="09/22/2019"></e-cell>
                            <e-cell value="12:30:02 PM"></e-cell>
                            <e-cell value="1,00,095.00"></e-cell>
                          </e-cells>
                        </e-row>
                      </e-rows>
                    </e-sheet>
                  </e-sheets>
                </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      style: { fontWeight : "bold", textAlign: "center" },
      validation: { type: 'WholeNumber', operator: 'NotEqualTo', value1:
'1'},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      listValidation: { type: 'List', value1: 'Digger, Digger, Cherrypicker'
},
      listValidation1: { type: 'List', value1: '50000,50000,45000' },
      validation1: { type: 'List', value1: 'Cherrypicker, JCB, Wheelbarrow'
},
      listValidation2:{ type: 'List', value1: '45000,90000,40' },
      listValidation3:{ type: 'List', value1: 'Kango, Ropes' },
      listValidation4:{ type: 'List', value1: '450, 95' },
      listValidation5:{ type: 'List', value1: 'JCB, Ropes, scaffolding' },
      listValidation6:{ type: 'List', value1: '90000, 95, 10000' }
    }
  },
  methods: {
   created: function () {
      var spreadsheet = this.$refs.spreadsheet;
       //Add Data validation to range.
      spreadsheet.addDataValidation({ type: 'TextLength' , operator:
'LessThanOrEqualTo' , value1: '4' }, 'A2:A5');
      spreadsheet.addDataValidation({ type: 'WholeNumber', operator:
'NotEqualTo', value1: '1' }, 'B2:B5');
      spreadsheet.addDataValidation({ type: 'Date', operator: 'NotEqualTo',
value1: '04/11/2019'}, 'F2:F5');
      spreadsheet.addDataValidation({ type: 'Time', operator: 'Between',
value1: '10:00:00 AM', value2: '11:00:00 AM' }, 'G2:G5');
      spreadsheet.addDataValidation({ type: 'Decimal', operator: 'LessThan',
value1: '100000.00'}, 'H2:H5');
      //Highlight Invalid Data.
      spreadsheet.addInvalidHighlight('A1:H5');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/number-format-cs1" %}

Limitations of Data validation

The following features have some limitations in Data Validation:

- Entire row data validation.
- Insert row between the data validation.
- Copy/paste with data validation.
- Delete cells between data validation applied range.

Copyright © 2001 -2024 Syncfusion Inc.

Clear

Clear feature helps you to clear the cell contents (formulas and data), formats (including number formats, conditional formats, and borders) in a spreadsheet. When you apply clear all, both the contents and the formats will be cleared simultaneously.

Apply Clear Feature

You can apply clear feature by using one of the following ways,

- Select the clear icon in the Ribbon toolbar under the Home Tab.
- Using the clear() method to clear the values.

Clear has the following types in the spreadsheet,

| Options | Uses |

|-----|------|

| Clear All | Used to clear all contents, formats, and hyperlinks. |

| Clear Formats | Used to clear the formats (including number formats, conditional formats, and borders) in a cell. |

| Clear Contents | Used to clear the contents (formulas and data) in a cell. |

| Clear Hyperlinks | Used to clear the hyperlink in a cell. |

Methods

Clear the cell contents and formats in the Spreadsheet document by using the clear method. The clear method has type and range as parameters. The following code example shows how to clear the cell contents and formats in the button click event.

APP.VUE

```
<template> <div>
<ejs-dropdownbutton :items='items' :select='itemSelect'>Clear</ejs-
dropdownbutton>
   <ejs-spreadsheet ref="spreadsheet" :created="created">
   <e-sheets>
          <e-sheet>
           <e-columns>
                     <e-column :width=88></e-column>
                     <e-column :width=88></e-column>
                     <e-column :width=160></e-column>
                     <e-column :width=100></e-column>
                     <e-column :width=150></e-column>
                   </e-columns>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
          </e-sheet>
        </e-sheets></ejs-spreadsheet></div>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { DropDownButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
```

```
import { orderData } from './data.js';
Vue.use(SpreadsheetPlugin);
Vue.use(DropDownButtonPlugin);
export default {
   data: () => {
    return {
      dataSource: orderData,
      items:[
        {
          text: "Clear All"
        },
        {
          text: "Clear Formats"
        },
        {
          text: "Clear Contents"
        },
        {
          text: "Clear Hyperlinks"
        }]
    }
  },
   methods: {
   created: function () {
        var spreadsheet = this.$refs.spreadsheet;
        spreadsheet.cellFormat({ fontWeight: 'bold', fontSize: '12pt'},
'A1:E1');
        spreadsheet.cellFormat({ color: '#10c469' }, 'B1:B10');
      },
     itemSelect: function(args) {
      var spreadsheet = this.$refs.spreadsheet;
      if (args.item.text === 'Clear All')
        spreadsheet.clear({ type: 'Clear All', range: 'D1:D10' }); // Clear
the content, formats and hyperlinks applied in the provided range.
      if (args.item.text === 'Clear Formats')
        spreadsheet.clear({ type: 'Clear Formats', range: 'B1:B10' }); //
Clear the formats applied in the provided range
      if (args.item.text === 'Clear Contents')
        spreadsheet.clear({ type: 'Clear Contents', range: 'A1:A10' }); //
Clear the content in the provided range
      if (args.item.text === 'Clear Hyperlinks')
        spreadsheet.clear({ type: 'Clear Hyperlinks', range: 'F2:F6' }); //
Clear the hyperlinks applied in the provided range
   }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/clear-cs1" %}

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Rows and columns
- Formatting
- Hyperlink
- Sorting
- Filtering

Editing in Vue Spreadsheet component

You can edit the contents of a cell directly in the cell or by typing in the formula bar. By default, the editing feature is enabled in the spreadsheet. Use the allowEditing property to enable or disable the editing feature.

Edit cell

You can start editing by one of the following ways,

- Double click a cell to start the edit mode.
- Press F2 key to edit the active cell.
- Use formula bar to perform editing.
- Use BACKSPACE or SPACE key to clear the cell content and start the edit mode.
- Using the startEdit method.

Save cell

If the cell is in editable state, you can save the edited cell by one of the following ways,

- Perform mouse click on any other cell rather than the current editing cell.
- Press Enter or Tab keys to save the edited cell content.
- Using the endEdit method.

Cancel editing

To cancel the editing without saving the changes, you can use one of the following ways,

- Press ESCAPE key, this will remove the editable state and update the unchanged cell content.
- Using the closeEdit method.

The following sample shows how to prevent the editing and cell save. Here E column prevent the editing by using cancel argument as true in cellEdit event. In D column, prevent saving the edited changes by using cancel argument as true in beforeCellSave and use closeEdit method in spreadsheet.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false" :cellEdit="cellEdit"
:beforeCellSave="beforeCellSave">
  <e-sheets>
            <e-sheet selectedRange="E11">
                    <e-ranges>
                      <e-range :dataSource="dataSource"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=120></e-column>
                      <e-column :width=180></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=120></e-column>
                      <e-column :width=120></e-column>
                    </e-columns>
                    <e-rows>
                      <e-row index="10">
                      <e-cells>
                      <e-cell index="3" value="Total Amount:"
style="fontStyle"></e-cell>
                      <e-cell formula="=SUM(E2:E10)"></e-cell>
                      </e-cells>
                      </e-row>
                    </e-rows>
                  </e-sheet>
                </e-sheets>
</ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
       fontStyle: { fontWeight: 'bold' }
    }
  },
  methods: {
  created: function () {
      var spreadsheet = this.$refs.spreadsheet;
     spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:E1');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'A2:A10');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'C2:C10');
        spreadsheet.numberFormat('$#,##0.00', 'D2:D10');
        spreadsheet.numberFormat('$#,##0.00', 'E2:E11');
      },
      cellEdit: function (args) {
       // Preventing the editing in 5th(Amount) column.
        if (args.address.includes('E')) { args.cancel = true; }
      },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    beforeCellSave: function (args) {
      var spreadsheet = this.$refs.spreadsheet;
         // Prevent saving the edited changes in 4th(Rate) column.
        if (args.address.includes('D')) {
            args.cancel = true;
            // Manually removes the editable state without saving the
changes. Use `endEdit` method if you want to save the changes.
            spreadsheet.closeEdit();
        }
      }
       },
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/editing-cs1" %}

Limitations

- Text overflow in cells is not supported in Editing.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Cell range
- Formatting
- Hyperlink
- Undo and Redo
- Unlock the particular cells in the protected sheet

Formulas in Vue Spreadsheet component

Formulas are used for calculating the data in a worksheet. You can refer the cell reference from same sheet or from different sheets.

Usage

You can set formula for a cell in the following ways,

- Using the formula property from cell, you can set the formula or expression to each cell at initial load.
- Set the formula or expression through data binding.
- You can set formula for a cell by editing.
- Using the updateCell method, you can set or update the cell formula.

Create User Defined Functions / Custom Functions

The Spreadsheet includes a number of built-in formulas. For your convenience, a list of supported formulas can be found here.

You can define and use an unsupported formula, i.e. a user defined/custom formula, in the spreadsheet by using the addCustomFunction function. Meanwhile, remember that you should define a user defined/custom formula whose results should only return a single value. If a user-defined/custom formula returns an array, it will be time-consuming to update adjacent cell values.

The following code example shows an unsupported formula in the spreadsheet.

APP.VUE

```
<template>
  <ejs-spreadsheet
    ref="spreadsheet"
    :created="created"
    :showRibbon="false"
    :showSheetTabs="false"
  >
    <e-sheets>
      <e-sheet>
        <e-ranges>
          <e-range :dataSource="dataSource" startCell="A2"></e-range>
        </e-ranges>
        <e-columns>
          <e-column :width="150"></e-column>
          <e-column :width="120"></e-column>
          <e-column :width="120"></e-column>
          <e-column :width="120"></e-column>
          <e-column :width="120"></e-column>
          <e-column :width="120"></e-column>
        </e-columns>
        <e-rows>
          <e-row :height="40" customHeight="true">
            <e-cells>
              <e-cell value="Monthly Expense" colSpan="5"></e-cell>
              <e-cell formula="=SUM(E2:E10)"></e-cell>
            </e-cells>
          </e-row>
          <e-row :height="30" customHeight="true"></e-row>
          <e-row :index="11">
            <e-cells>
              <e-cell value="Totals"></e-cell>
              <e-cell formula="=SUM(B3:B11)"></e-cell>
              <e-cell formula="=SUM(C3:C11)"></e-cell>
              <e-cell formula="=SUM(D3:D11)"></e-cell>
            </e-cells>
          </e-row>
```

```
            <e-row>
              <e-cells>
                <e-cell
                  :index="1"
                  value="Number of Categories"
                  :colSpan="2"
                ></e-cell>
                <e-cell index="3" formula="=COUNTA(A3:A11)"></e-cell>
              </e-cells>
            </e-row>
            <e-row>
              <e-cells>
                <e-cell :index="1" value="Average Spend" :colSpan="2"></e-
cell>
                <e-cell
                  index="3"
                  formula="=AVERAGE(B3:B11)"
                  format="$#,##0"
                ></e-cell>
              </e-cells>
            </e-row>
            <e-row>
              <e-cells>
                <e-cell :index="1" value="Min Spend" :colSpan="2"></e-cell>
                <e-cell index="3" formula="=MIN(B3:B11)" format="$#,##0"></e-
cell>
              </e-cells>
            </e-row>
            <e-row>
              <e-cells>
                <e-cell :index="1" value="Max Spend" :colSpan="2"></e-cell>
                <e-cell index="3" formula="=Max(B3:B11)" format="$#,##0"></e-
cell>
              </e-cells>
            </e-row>
          </e-rows>
        </e-sheet>
      </e-sheets>
    </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
  data: () => {
    return {
      dataSource: data,
    };
  },
  methods: {
    // Custom function to calculate percentage between two cell values.
    calculatePercentage(firstCell, secondCell) {
      return Number(firstCell) / Number(secondCell);
    },
    // Custom function to calculate round down for values.
```

```
    roundDownHandler(value, digit) {
      let multiplier = Math.pow(10, digit);
      return Math.floor(value * multiplier) / multiplier;
    },
    created: function () {
      var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.cellFormat(
        { fontWeight: "bold", textAlign: "center" },
        "A2:F2"
      );
      spreadsheet.numberFormat("$#,##0", "B3:D12");
      spreadsheet.numberFormat("0%", "E3:E12");
      // Adding custom function for calculating the percentage between two
cells.
      spreadsheet.addCustomFunction(this.calculatePercentage, "PERCENTAGE");
      // Adding custom function for calculating round down for the value.
      spreadsheet.addCustomFunction(this.roundDownHandler, "ROUNDDOWN");
      // Calculate percentage using custom added formula in E12 cell.
      spreadsheet.updateCell({ formula: "=PERCENTAGE(C12,D12)" }, "E12");
      // Calculate round down for average values using custom added formula
in F12 cell.
      spreadsheet.updateCell({ formula: "=ROUNDDOWN(F11,1)" }, "F12");
    },
  },
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/formula-cs1" %}

Second, if you want to directly compute any formula or expression, you can use the computeExpression method. This method will work for both built-in and used-defined/custom formula.

The following code example shows how to use computeExpression method in the spreadsheet.

APP.VUE

```
<template>
  <ejs-spreadsheet
    ref="spreadsheet"
    :created="created"
    :showRibbon="false"
    :showSheetTabs="false"
  >
    <e-sheets>
```

```
      <e-sheet>
        <e-ranges>
          <e-range :dataSource="dataSource" startCell="A2"></e-range>
        </e-ranges>
        <e-columns>
          <e-column :width="150"></e-column>
          <e-column :width="120"></e-column>
          <e-column :width="120"></e-column>
          <e-column :width="120"></e-column>
          <e-column :width="120"></e-column>
        </e-columns>
        <e-rows>
          <e-row :height="40" customHeight="true">
            <e-cells>
              <e-cell value="Monthly Expense" colSpan="5"></e-cell>
              <e-cell formula="=SUM(E2:E10)"></e-cell>
            </e-cells>
          </e-row>
          <e-row :height="30" customHeight="true"></e-row>
          <e-row :index="11">
            <e-cells>
              <e-cell value="Totals"></e-cell>
              <e-cell formula="=SUM(B3:B11)"></e-cell>
              <e-cell formula="=SUM(C3:C11)"></e-cell>
              <e-cell formula="=SUM(D3:D11)"></e-cell>
            </e-cells>
          </e-row>
          <e-row>
            <e-cells>
              <e-cell
                :index="1"
                value="Number of Categories"
                :colSpan="2"
              ></e-cell>
              <e-cell index="3" formula="=COUNTA(A3:A11)"></e-cell>
            </e-cells>
          </e-row>
          <e-row>
            <e-cells>
              <e-cell :index="1" value="Average Spend" :colSpan="2"></e-
cell>
              <e-cell
                index="3"
                formula="=AVERAGE(B3:B11)"
                format="$#,##0"
              ></e-cell>
            </e-cells>
          </e-row>
          <e-row>
            <e-cells>
              <e-cell :index="1" value="Min Spend" :colSpan="2"></e-cell>
              <e-cell index="3" formula="=MIN(B3:B11)" format="$#,##0"></e-
cell>
            </e-cells>
          </e-row>
          <e-row>
            <e-cells>
```

```
              <e-cell :index="1" value="Max Spend" :colSpan="2"></e-cell>
              <e-cell index="3" formula="=Max(B3:B11)" format="$#,##0"></e-
cell>
            </e-cells>
          </e-row>
        </e-rows>
      </e-sheet>
    </e-sheets>
  </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
  data: () => {
    return {
       dataSource: data,
    };
  },
  methods: {
    // Custom function to calculate percentage between two cell values.
    calculatePercentage(firstCell, secondCell) {
      return Number(firstCell) / Number(secondCell);
    },
    created: function () {
      var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.cellFormat(
        { fontWeight: "bold", textAlign: "center" },
        "A2:F2"
      );
      spreadsheet.numberFormat("$#,##0", "B3:D12");
      spreadsheet.numberFormat("0%", "E3:E12");
      // Adding custom function for calculating the percentage between two
cells.
      spreadsheet.addCustomFunction(this.calculatePercentage, "PERCENTAGE");
      // Calculate percentage using custom added formula in E11 cell.
      spreadsheet.updateCell({ formula: "=PERCENTAGE(C11,D11)" }, "E11");
      // Calculate expressions using computeExpression in E10 cell.
      spreadsheet.updateCell(
        { value: spreadsheet.computeExpression("C10/D10") },
        "E10"
      );
      // Calculate custom formula values using computeExpression in E12
cell.
      spreadsheet.updateCell(
        {
          value: spreadsheet.computeExpression("=PERCENTAGE(C12,D12)"),
        },
        "E12"
      );
      // Calculate SUM (built-in) formula values using computeExpression in
D12 cell.
      spreadsheet.updateCell(
        {
          value: spreadsheet.computeExpression("=SUM(D3:D11)"),
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        },
        "D12"
      );
    },
  },
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/formula-cs2" %}

Formula bar

Formula bar is used to edit or enter cell data in much easier way. By default, the formula bar is enabled in the spreadsheet. Use the showFormulaBar property to enable or disable the formula bar.

Named Ranges

You can define a meaningful name for a cell range and use it in the formula for calculation. It makes your formula much easier to understand and maintain. You can add named ranges to the Spreadsheet in the following ways,

- Using the definedNames collection, you can add multiple named ranges at initial load.
- Use the addDefinedName method to add a named range dynamically.
- You can remove an added named range dynamically using the removeDefinedName method.
- Select the range of cells, and then enter the name for the selected range in the name box.

The following code example shows the usage of named ranges support.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :created="created" :showRibbon="false"
:beforeDataBound="beforeDataBound"
  :showSheetTabs="false">
  <e-definednames>
                  <e-definedname name="Categories" refersTo="=Budget
Details!A3:A11"></e-definedname>
                  <e-definedname name="MonthlySpendings" refersTo="=Budget
Details!B3:B11"></e-definedname>
                  <e-definedname name="AnnualSpendings" refersTo="=Budget
Details!C3:C11"></e-definedname>
                </e-definednames>
  <e-sheets>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    <e-sheet name="Budget Details">
      <e-ranges>
        <e-range :dataSource="dataSource" startCell="A2"></e-range>
      </e-ranges>
      <e-columns>
        <e-column :width=150></e-column>
        <e-column :width=120></e-column>
        <e-column :width=120></e-column>
        <e-column :width=120></e-column>
        <e-column :width=120></e-column>
      </e-columns>
      <e-rows>
        <e-row :height=40 customHeight="true">
          <e-cells>
            <e-cell value="Monthly Expense" colSpan=5></e-cell>
            <e-cell formula="=SUM(E2:E10)"></e-cell>
          </e-cells>
        </e-row>
        <e-row :height=30 customHeight="true"></e-row>
        <e-row :index=11>
          <e-cells>
            <e-cell value="Totals"></e-cell>
            <e-cell formula="=SUM(MonthlySpendings)"></e-cell>
            <e-cell formula="=SUM(AnnualSpendings)"></e-cell>
            <e-cell formula="=SUM(LastYearSpendings)"></e-cell>
          </e-cells>
        </e-row>
        <e-row>
          <e-cells>
            <e-cell :index=1 value="Number of Categories" :colSpan=2 ></e-
cell>
            <e-cell index=3 formula="=COUNTA(Categories)"></e-cell>
          </e-cells>
        </e-row>
        <e-row>
          <e-cells>
            <e-cell :index=1 value="Average Spend" :colSpan=2 ></e-cell>
            <e-cell index=3 formula="=AVERAGE(MonthlySpendings)"
format="$#,##0"></e-cell>
          </e-cells>
        </e-row>
        <e-row>
          <e-cells>
            <e-cell :index=1 value="Min Spend" :colSpan=2 ></e-cell>
            <e-cell index=3 formula="=MIN(MonthlySpendings)"
format="$#,##0"></e-cell>
          </e-cells>
        </e-row>
        <e-row>
          <e-cells>
            <e-cell :index=1 value="Max Spend" :colSpan=2></e-cell>
            <e-cell index=3 formula="=Max(MonthlySpendings)"
format="$#,##0"></e-cell>
          </e-cells>
        </e-row>
      </e-rows>
    </e-sheet>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  </e-sheets>
</ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
    }
  },
 methods: {
    // Custom function to calculate percentage between two cell values.
    calculatePercentage(firstCell, secondCell) {
      return Number(firstCell) / Number(secondCell);
    },
    beforeDataBound: function () {
      var spreadsheet = this.$refs.spreadsheet;
    // Adding name dynamically for `last year spending` and `percentage
change` ranges.
    spreadsheet.addDefinedName({ name: 'LastYearSpendings', refersTo:
'=D3:D11' });
    spreadsheet.addDefinedName({ name: 'PercentageChange', refersTo:
'=E3:E11' });
    },
    created: function () {
        var spreadsheet = this.$refs.spreadsheet;
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A2:E2');
        spreadsheet.cellFormat( { fontWeight: 'bold', textAlign: 'right' },
'B13:B16');
        spreadsheet.cellFormat(  { fontStyle: 'italic', fontWeight: 'bold'
}, 'A12');
        spreadsheet.cellFormat(  { textAlign: 'center', fontWeight: 'bold',
verticalAlign: 'middle', fontStyle: 'italic', fontSize: '15pt' }, 'A1');
        spreadsheet.numberFormat('$#,##0', 'B3:D12');
        spreadsheet.numberFormat('0%', 'E3:E12');
        // Adding custom function for calculating the percentage between two
cells.
        spreadsheet.addCustomFunction(this.calculatePercentage,
'PERCENTAGE');
        // Calculate percentage using custom added formula in E12 cell.
        spreadsheet.updateCell({ formula: '=PERCENTAGE(C12,D12)' }, 'E12');
        }
      }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
  @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/defined-name-cs1" %}

Supported Formulas

The list of supported formulas can be find in following link.

Formula Error Dialog

If you enter an invalid formula in a cell, an error dialog with an error message will appear. For instance, a formula with the incorrect number of arguments, a formula without parenthesis, etc.

| Error Message | Reason |

|-------|---------|

| We found that you typed a formula with an invalid arguments | Occurs when passing an argument even though it wasn't needed. |

| We found that you typed a formula with an empty expression | Occurs when passing an empty expression in the argument. |

| We found that you typed a formula with one or more missing opening or closing parenthesis | Occurs when an open parenthesis or a close parenthesis is missing. |

| We found that you typed a formula which is improper | Occurs when passing a single reference but a range was needed. |

| We found that you typed a formula with a wrong number of arguments | Occurs when the required arguments were not passed. |

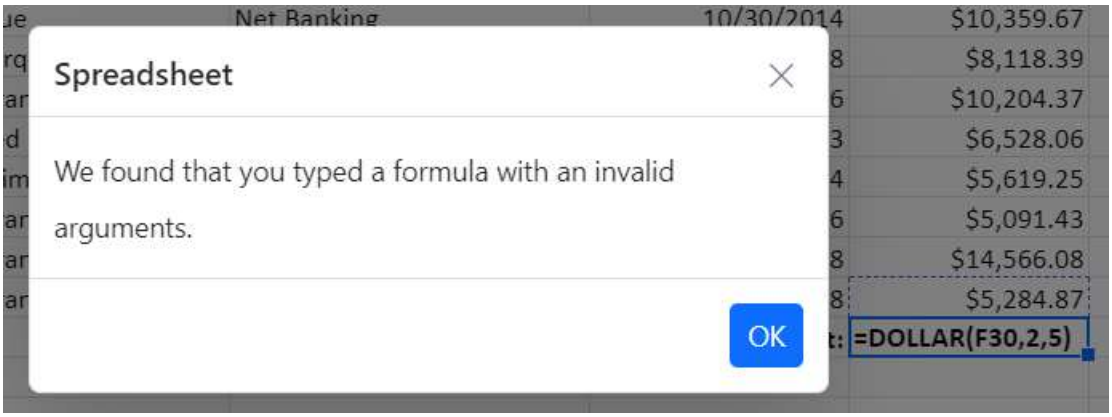
| We found that you typed a formula which requires 3 arguments | Occurs when the required 3 arguments were not passed. |

| We found that you typed a formula with a mismatched quotes | Occurs when passing an argument with mismatched quotes. |

| We found that you typed a formula with a circular reference | Occurs when passing a formula with circular cell reference. |

| We found that you typed a formula which is invalid | Except in the cases mentioned above, all other errors will fall into this broad category. |

Copyright © 2001 -2024 Syncfusion Inc.



Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Editing
- Formatting
- Open
- Save

Formatting in Vue Spreadsheet component

Formatting options make your data easier to view and understand. The different types of formatting options in the Spreadsheet are,

- Number Formatting
- Text Formatting
- Cell Formatting

Number Formatting

Number formatting provides a type for your data in the Spreadsheet. Use the allowNumberFormatting property to enable or disable the number formatting option in the Spreadsheet. The different types of number formatting supported in Spreadsheet are,

Types	Format
General(default)	NA
Number	0.00
Currency	$#,##0.00
Accounting	*($ #,##0.00);($ (#,##0.00);*($* "-"??*);(@_)*
ShortDate	mm-dd-yyyy
LongDate	dddd, mmmm dd, yyyy

Copyright © 2001 -2024 Syncfusion Inc.

| Time | h:mm:ss AM/PM |

| Percentage | 0.00% |

| Fraction | # ?/? |

| Scientific |0.00E+00 |

| Text | @ |

Number formatting can be applied in following ways,

- Using the format property in cell, you can set the desired format to each cell at initial load.
- Using the numberFormat method, you can set the number format to a cell or range of cells.
- Selecting the number format option from ribbon toolbar.

The following code example shows the number formatting in cell data.

APP.VUE

```
<template>
 <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false"
              :showRibbon="false" :showSheetTabs="false"
:allowInsert="false" :allowDelete="false">
                <e-sheets>
                  <e-sheet :showGridLines="false">
                    <e-ranges>
                      <e-range :dataSource="dataSource" startCell="A2"></e-
range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=140></e-column>
                      <e-column :width=140></e-column>
                      <e-column :width=160></e-column>
                      <e-column :width=160></e-column>
                      <e-column :width=160></e-column>
                      <e-column :width=120></e-column>
                    </e-columns>
                    <e-rows>
                      <e-row :height=35 customHeight="true">
                        <e-cells>
                          <e-cell value="Sales Team Summary" colSpan=6></e-
cell>
                        </e-cells>
                      </e-row>
                      <e-row :index=10>
                        <e-cells>
                          <e-cell index=1 value="Total:"></e-cell>
                          <e-cell formula="=SUM(C3:C10)"
format="$#,##0.00"></e-cell>
                          <e-cell formula="=SUM(D3:D10)" format='_($*
#,##0.00_);_($* (#,##0.00);_($* "-"??_);_(@_)'></e-cell>
                          <e-cell formula="=SUM(E3:E10)" format='_($*
#,##0.00_);_($* (#,##0.00);_($* "-"??_);_(@_)'></e-cell>
                        </e-cells>
                      </e-row>
```

```
                    </e-rows>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.cellFormat({ fontWeight: 'bold', fontSize: '12pt',
backgroundColor: '#279377', textAlign: 'center',
          color: '#ffffff', borderBottom: '1px solid #e0e0e0' }, 'A2:F2');
        spreadsheet.cellFormat({ borderTop: '1px solid #e0e0e0',
backgroundColor: '#EEEEEE' }, 'A11:F11');
        spreadsheet.cellFormat({ fontWeight: 'bold', fontStyle: 'italic' },
'B11');
        spreadsheet.cellFormat({ verticalAlign: 'middle', textAlign:
'center', fontSize: '16pt', fontWeight: 'bold', border: '1px solid #e0e0e0',
backgroundColor: '#EEEEEE', color: '#279377' }, 'A1');
        spreadsheet.setBorder({ border: '1px solid #e0e0e0' }, 'A2:F11',
'Outer');
        // Applied Accounting format to the cells from C3 to E10 range.
        spreadsheet.numberFormat('_($* #,##0.00_);_($* (#,##0.00);_($* "-
"??_);_(@_)', 'C3:E10');
        // Applied Percentage format to the cells from C3 to E11 range.
        spreadsheet.numberFormat('0%', 'F3:F10');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/number-format-cs2" %}

Copyright © 2001 -2024 Syncfusion Inc.

Limitations of Number Formatting

- Custom number formatting is not supported.

Text and cell formatting

Text and cell formatting enhances the look and feel of your cell. It helps to highlight a particular cell or range of cells from a whole workbook. You can apply formats like font size, font family, font color, text alignment, border etc. to a cell or range of cells. Use the allowCellFormatting property to enable or disable the text and cell formatting option in Spreadsheet. You can set the formats in following ways,

- Using the `style` property, you can set formats to each cell at initial load.
- Using the cellFormat method, you can set formats to a cell or range of cells.
- You can also apply by clicking the desired format option from the ribbon toolbar.

Fonts

Various font formats supported in the spreadsheet are font-family, font-size, bold, italic, strike-through, underline and font color.

Text Alignment

You can align text in a cell either vertically or horizontally using the `textAlign` and `verticalAlign` property.

Indents

To enhance the appearance of text in a cell, you can change the indentation of a cell content using `textIndent` property.

Fill color

To highlight cell or range of cells from whole workbook you can apply background color for a cell using `backgroundColor` property.

Borders

You can add borders around a cell or range of cells to define a section of worksheet or a table. The different types of border options available in the spreadsheet are,

| Types | Actions |

|-------|---------|

| Top Border | Specifies the top border of a cell or range of cells.|

| Left Border | Specifies the left border of a cell or range of cells.|

| Right Border | Specifies the right border of a cell or range of cells.|

| Bottom Border | Specifies the bottom border of a cell or range of cells.|

| No Border | Used to clear the border from a cell or range of cells.|

| All Border | Specifies all border of a cell or range of cells.|

| Horizontal Border | Specifies the top and bottom border of a cell or range of cells.|

| Vertical Border | Specifies the left and right border of a cell or range of cells.|

| Outside Border | Specifies the outside border of a range of cells.|

Copyright © 2001 -2024 Syncfusion Inc.

| Inside Border | Specifies the inside border of a range of cells.|

You can also change the color, size, and style of the border. The size and style supported in the spreadsheet are,

| Types | Actions |

|-------|---------|

| Thin | Specifies the 1px border size (default).|

| Medium | Specifies the 2px border size.|

| Thick | Specifies the 3px border size.|

| Solid | Used to create the solid border (default).|

| Dashed | Used to create the dashed border.|

| Dotted | Used to create the dotted border.|

| Double | Used to create the double border.|

Borders can be applied in the following ways,

- Using the border, borderLeft, borderRight, borderBottom properties, you can set the desired border to each cell at initial load.
- Using the setBorder method, you can set various border options to a cell or range of cells.
- Selecting the border options from ribbon toolbar.

The following code example shows the style formatting in text and cells of the spreadsheet.

APP.VUE

```
<template>
    <ejs-spreadsheet ref= "spreadsheet" :created="created"
:showFormulaBar="false" :showRibbon="false"
                :showSheetTabs="false" :allowInsert="false"
:allowDelete="false" :allowEditing="false">
                <e-sheets>
                  <e-sheet :showGridLines="false">
                    <e-ranges>
                      <e-range :dataSource="dataSource" startCell="A2"></e-
range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=100></e-column>
                      <e-column :width=200></e-column>
                      <e-column :width=110></e-column>
                      <e-column :width=140></e-column>
                      <e-column :width=90></e-column>
                    </e-columns>
                    <e-rows>
                      <e-row :height=35 customHeight="true">
                        <e-cells>
                          <e-cell value="Order Summary" colSpan=5></e-cell>
                        </e-cells>
                      </e-row>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                        <e-row :height=30>
                          <e-cells>
                            <e-cell index=2></e-cell>
                          </e-cells>
                        </e-row>
                        <e-row :height=30></e-row>
                        <e-row :height=30></e-row>
                        <e-row :height=30></e-row>
                        <e-row :height=30></e-row>
                        <e-row :height=30></e-row>
                        <e-row :height=30></e-row>
                        <e-row :height=30></e-row>
                        <e-row :height=30></e-row>
                        <e-row :height=30></e-row>
                      </e-rows>
                    </e-sheet>
                  </e-sheets>
                </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
    }
  },
  methods: {
  created: function () {
       var spreadsheet = this.$refs.spreadsheet;
      // Setting common styles to table header cells
        spreadsheet.cellFormat({ fontWeight: 'bold', fontSize: '12pt',
backgroundColor: '#279377', color: '#ffffff' }, 'A2:E2');
        // Setting common styles to whole table cells
        spreadsheet.cellFormat({ verticalAlign: 'middle', fontFamily:
'Axettac Demo' }, 'A2:E12');
        spreadsheet.cellFormat({textAlign: 'right'}, 'B2');
        spreadsheet.cellFormat({ fontFamily: 'Axettac Demo', verticalAlign:
'middle', textAlign: 'center', fontSize: '18pt', fontWeight: 'bold', color:
'#279377', border: '1px solid #e0e0e0' }, 'A1');
        // Column wise styles setting
        spreadsheet.cellFormat({ textAlign: 'center' }, 'A2:A12');
        // Setting text-indent to 2 and 4 column
        var style = { textAlign: 'left', textIndent: '8pt' };
        spreadsheet.cellFormat(style, 'B2:B12');
        spreadsheet.cellFormat(style, 'D2:D12');
        spreadsheet.cellFormat({ fontStyle: 'italic', textAlign: 'right' },
'C3:C12');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'E2:E12');
        // Applied border to range of cells using 'setBorder' method
        spreadsheet.setBorder({ borderLeft: '1px solid #e0e0e0',
borderRight: '1px solid #e0e0e0' }, 'A2:E2');
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        spreadsheet.setBorder({ border: '1px solid #e0e0e0' }, 'A4:E11',
'Horizontal');
        spreadsheet.setBorder({ border: '1px solid #e0e0e0' }, 'A3:E12',
'Outer');
        spreadsheet.cellFormat({ color: '#10c469', textDecoration: 'line-
through' }, 'E3:E4');
        spreadsheet.cellFormat({ color: '#10c469', textDecoration: 'line-
through' }, 'E9');
        spreadsheet.cellFormat({ color: '#10c469', textDecoration: 'line-
through' }, 'E12');
        spreadsheet.cellFormat({ color: '#FFC107', textDecoration:
'underline' }, 'E5');
        spreadsheet.cellFormat({ color: '#FFC107', textDecoration:
'underline' }, 'E8');
        spreadsheet.cellFormat({ color: '#FFC107', textDecoration:
'underline' }, 'E11');
        spreadsheet.cellFormat({ color: '#62c9e8' }, 'E6');
        spreadsheet.cellFormat({ color: '#62c9e8' }, 'E10');
        spreadsheet.cellFormat({ color: '#ff5b5b' }, 'E7');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/cell-format-cs1" %}

Limitations of Formatting

The following features are not supported in Formatting:

- Insert row/column between the formatting applied cells.
- Formatting support for row/column.

Conditional Formatting

Conditional formatting helps you to format a cell or range of cells based on the conditions applied. You can enable or disable conditional formats by using the allowConditionalFormat property.

* The default value for the `allowConditionalFormat` property is `true`.

Apply Conditional Formatting

You can apply conditional formatting by using one of the following ways,

- Select the conditional formatting icon in the Ribbon toolbar under the Home Tab.
- Using the conditionalFormat() method to define the condition.
- Using the `conditionalFormats` in sheets model.

Conditional formatting has the following types in the spreadsheet,

Highlight cells rules

Highlight cells rules option in the conditional formatting enables you to highlight cells with a preset color depending on the cell's value.

The following options can be given for the highlight cells rules as type,

* 'GreaterThan', 'LessThan', 'Between', 'EqualTo', 'ContainsText', 'DateOccur', 'Duplicate', 'Unique'.

The following preset colors can be used for formatting styles,

* `“RedFT”` - Light Red Fill with Dark Red Text,

* `“YellowFT”` - Yellow Fill with Dark Yellow Text,

* `“GreenFT”` - Green Fill with Dark Green Text,

* `“RedF”` - Red Fill,

* `“RedT”` - Red Text.

Top bottom rules

Top bottom rules option in the conditional formatting allows you to apply formatting to the cells that satisfy a statistical condition with other cells in the range.

The following options can be given for the top bottom rules as type,

* 'Top10Items', 'Bottom10Items', 'Top10Percentage', 'Bottom10Percentage', 'BelowAverage', 'AboveAverage'.

Data Bars

You can apply data bars to represent the data graphically inside a cell. The longest bar represents the highest value and the shorter bars represent the smaller values.

The following options can be given for the data bars as type,

* 'BlueDataBar', 'GreenDataBar', 'RedDataBar', 'OrangeDataBar', 'LightBlueDataBar', 'PurpleDataBar'.

Color Scales

Using color scales, you can format your cells with two or three colors, where different color shades represent the different cell values. In the Green-Yellow-Red(GYR) Color Scale, the cell that holds the minimum value is colored as red. The cell that holds the median is colored as yellow, and the cell that holds the maximum value is colored as green. All other cells are colored proportionally.

The following options can be given for the color scales as type,

* 'GYRColorScale', 'RYGColorScale', 'GWRColorScale', 'RWGColorScale', 'BWRColorScale', 'RWBColorScale', 'WRColorScale', 'RWColorScale', 'GWColorScale', 'WGColorScale', 'GYColorScale', 'YGColorScale'.

Copyright © 2001 -2024 Syncfusion Inc.

Icon Sets

Icon sets will help you to visually represent your data with icons. Every icon represents a range of values. In the Three Arrows(colored) icon, the green arrow icon represents the values greater than 67%, the yellow arrow icon represents the values between 33% to 67%, and the red arrow icon represents the values less than 33%.

The following options can be given for the icon sets as type,

* 'ThreeArrows', 'ThreeArrowsGray', 'FourArrowsGray', 'FourArrows', 'FiveArrowsGray', 'FiveArrows', 'ThreeTrafficLights1', 'ThreeTrafficLights2', 'ThreeSigns', 'FourTrafficLights', 'FourRedToBlack', 'ThreeSymbols', 'ThreeSymbols2', 'ThreeFlags', 'FourRating', 'FiveQuarters', 'FiveRating', 'ThreeTriangles', 'ThreeStars', 'FiveBoxes'.

Custom Format

Using the custom format for conditional formatting you can set cell styles like color, background color, font style, font weight, and underline.

In the MAY and JUN columns, we have applied conditional formatting custom format.

* In the Conditional format, custom format supported for Highlight cell rules and Top bottom rules.

Clear Rules

You can clear the defined rules by using one of the following ways,

- Using the “Clear Rules” option in the Conditional Formatting button of HOME Tab in the ribbon to clear the rule from selected cells.
- Using the clearConditionalFormat() method to clear the defined rules.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false">
  <e-sheets>
    <e-sheet name="Car Sales Record">
      <e-ranges>
        <e-range :dataSource="dataSource"></e-range>
      </e-ranges>
      <e-columns>
        <e-column :width=120></e-column>
      </e-columns>
      <e-conditionalformats>
        <e-conditionalformat type="GreaterThan", cFColor="RedFT",
value="700", range="B2:B9" ></e-conditionalformat>
        <e-conditionalformat type="Bottom10Items", cFColor="YellowFT",
value="4", range="C2:C9"></e-conditionalformat>
        <e-conditionalformat type="BlueDataBar", range="D2:D9"></e-
conditionalformat>
      </e-conditionalformats>
    </e-sheet>
  </e-sheets>
</ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { conditionalFormatData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: conditionalFormatData,
    }
  },
  methods: {
  created: function () {
      var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:N1');
      spreadsheet.conditionalFormat({ type: "RYGColorScale", range: 'E2:E9'
});
      spreadsheet.conditionalFormat({ type: "ThreeArrows", range: 'H2:H9'
});
    //Custom format
        spreadsheet.conditionalFormat({ type: 'Top10Items', value: '1',
            format: { style: { color: '#ffffff', backgroundColor: '#009999',
fontWeight: 'bold'}}, range: 'F2:F9' });
        spreadsheet.conditionalFormat({ type: 'Bottom10Items', value: '1',
            format: { style: { color: '#ffffff', backgroundColor: '#c68d53',
fontWeight: 'bold'}}, range: 'G2:G9' });
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/conditional-formatting-cs1" %}

Limitations of Conditional formatting

The following features have some limitations in Conditional Formatting:

- Insert row/column between the conditional formatting.
- Conditional formatting with formula support.
- Copy and paste the conditional formatting applied cells.
- Custom rule support.

Copyright © 2001 -2024 Syncfusion Inc.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Rows and columns
- Hyperlink
- Sorting
- Filtering
- Ribbon customization

Freeze pane in Vue Spreadsheet component

Freeze Panes helps you to keep particular rows or columns visible when scrolling the sheet content in the spreadsheet. You can specify the number of frozen rows and columns using the frozenRows and frozenColumns properties inside the Sheet property.

Apply freezepanes on UI

User Interface:

In the active spreadsheet, click the cell where you want to create freeze panes. Freeze panes can be done in any of the following ways:

- Select the View tab in the Ribbon toolbar and choose the Freeze Panes item.
- Use the freezePanes method programmatically.

FrozenRows

It allows you to keep a certain number of rows visible while scrolling vertically through the rest of the worksheet.

User Interface:

In the active spreadsheet, select the cell where you want to create frozen rows. Frozen rows can be done in any one of the following ways:

- Select the View tab in the Ribbon toolbar and choose the Freeze Rows item.
- You can specify the number of frozen rows using the frozenRows property inside the Sheet property.

FrozenColumns

It allows you to keep a certain number of columns visible while scrolling horizontally through the rest of the worksheet.

User Interface:

In the active spreadsheet, select the cell where you want to create frozen columns. Frozen columns can be done in any one of the following ways:

- Select the View tab in the Ribbon toolbar and choose the Freeze Columns item.

Copyright © 2001 -2024 Syncfusion Inc.

- You can specify the number of frozen columns using the frozenColumns property inside the Sheet property.

In this demo, the frozenColumns is set as '2', and the frozenRows is set as '2'. Hence, the two columns on the left and the top two rows are frozen.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet">
   <e-sheets>
          <e-sheet :frozenRows=2  :frozenColumns=2 >
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
             <e-columns>
                      <e-column :width=120></e-column>
                      <e-column :width=180></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=120></e-column>
                      <e-column :width=120></e-column>
              </e-columns>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: defaultData,
    }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/freezepane-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Limitations

Here, we have listed out the limitations with Freeze Panes feature.

- Merging the cells between freeze and unfreeze area.
- If images and charts are added inside the freeze area cells, their portion in the unfreeze area will not move when scrolling.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Sorting
- Filtering
- Undo Redo

Context menu in Vue Spreadsheet component

Context Menu is used to improve user interaction with Spreadsheet using the popup menu. This will open when right-clicking on Cell/Column Header/Row Header/ Pager in the Spreadsheet. You can use enableContextMenu property to enable/disable context menu.

The default value for the `enableContextMenu` property is `true`.

Context Menu Items in Row Cell

Please find the table below for default context menu items and their actions.

| Context Menu items | Action |

|-------|---------|

| Cut | Cut the selected cells data to the clipboard, you can select a cell where you want to move the data. |

| Copy | Copy the selected cells data to the clipboard, so that you can paste it to somewhere else. |

| Paste | Paste the data from clipboard to spreadsheet. |

| Paste Special | `Values` - Paste the data values from clipboard to spreadsheet. `Formats` - Paste the data formats from clipboard to spreadsheet. |

| Filter | Perform filtering to the selected cells based on an active cell's value. |

| Sort | Perform sorting to the selected range of cells by ascending or descending. |

| Hyperlink | Create a link in the spreadsheet to navigate to web links or cell reference within the sheet or other sheets in the Spreadsheet. |

Context Menu Items in Row Header / Column Header

Please find the table below for default context menu items and their actions.

| Context Menu items | Action |

|-------|---------|

Copyright © 2001 -2024 Syncfusion Inc.

| Cut | Cut the selected row/column header data to the clipboard, you can select a cell where you want to move the data. |

| Copy | Copy the selected row/column header data to the clipboard, so that you can paste it to somewhere else. |

| Paste | Paste the data from clipboard to spreadsheet. |

| Paste Special | Values - Paste the data values from clipboard to spreadsheet. Formats - Paste the data formats from clipboard to spreadsheet. |

| Insert Columns | Insert new rows or columns into the worksheet. |

| Delete Columns | Delete existing rows or columns from the worksheet. |

| Hide Columns | Hide the rows and columns. |

| UnHide Columns | Show the hidden rows and columns. |

Context Menu Items in Pager

Please find the table below for default context menu items and their actions.

| Context Menu items | Action |

|-------|---------|

| Insert | Insert a new worksheet in front of an existing worksheet in the spreadsheet. |

| Delete | Delete the selected worksheet from the spreadsheet. |

| Rename | Rename the selected worksheet. |

| Protect Sheet | Prevent unwanted changes from others by limiting their ability to edit. |

| Hide |Hide the selected worksheet. |

Context Menu Customization

You can perform the following context menu customization options in the spreadsheet

- Add Context Menu Items
- Remove Context Menu Items
- Enable/Disable Context Menu Items

Add Context Menu Items

You can add the custom items in context menu using the addContextMenuItems in contextmenuBeforeOpen event

In this demo, Custom Item is added after the Paste item in the context menu.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet"
:contextMenuBeforeOpen="contextMenuBeforeOpen"></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
Vue.use(SpreadsheetPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  methods: {
    contextMenuBeforeOpen: function (args) {
      var spreadsheet = this.$refs.spreadsheet;
      if (args.element.id === spreadsheet.$el.id + '_contextmenu') {
        // To add context menu items.
        spreadsheet.addContextMenuItems([{ text: 'Custom Item' }], 'Paste
Special', false); //To pass the items, Item before / after that the element
to be inserted, Set false if the items need to be inserted before the text.
      }
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/contextmenu/addContextMenu-cs1" %}

Remove Context Menu Items

You can remove the items in context menu using the removeContextMenuItems in contextmenuBeforeOpen event

In this demo, Insert Column item has been removed from the row/column header context menu.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet"
:contextMenuBeforeOpen="contextMenuBeforeOpen"></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
Vue.use(SpreadsheetPlugin);
export default {
  methods: {
  contextMenuBeforeOpen: function () {
        var spreadsheet = this.$refs.spreadsheet;
        // To remove context menu items.
         spreadsheet.removeContextMenuItems(["Insert Column"], false);
//Items that needs to be removed, Set `true` if the given `text` is a unique
id.
      }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/contextmenu/addContextMenu-cs2" %}

Enable/Disable Context Menu Items

You can enable/disable the items in context menu using the enableContextMenuItems in contextmenuBeforeOpen event

In this demo, Rename item is disabled in the pager context menu.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet"
:contextMenuBeforeOpen="contextMenuBeforeOpen"></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
Vue.use(SpreadsheetPlugin);
export default {
  methods: {
  contextMenuBeforeOpen: function () {
        var spreadsheet = this.$refs.spreadsheet;
        //To enable / disable context menu items.
        spreadsheet.enableContextMenuItems(['Rename'], false, false); //
Contextmenu Items that needs to be enabled / disabled, Set true / false to
enable / disable the menu items, Set true if the given text is a unique id.
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
```

```
  @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
  @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/contextmenu/addContextMenu-cs3" %}

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Worksheet
- Rows and columns

Template in Vue Spreadsheet component

Cell Template is used for adding HTML elements into Spreadsheet. You can add the cell template in spreadsheet by using the `template` property and specify the address using the `address` property inside the `ranges` property. You can customize the Html elements similar to Syncfusion components (TextBox, DropDownList, RadioButton, MultiSelect, DatePicker etc) by using the `beforeCellRender` event. In this demo, Cell template is applied to `C2:C9` and instantiated with Html input components like TextBox, RadioButton, TextArea. You need to bind the events to perform any operations through HTML elements or Syncfusion components. Here, we have added `change` event in to the MultiSelect control, and we have updated the selected data into the spreadsheet cell through that change event.

<!-- `

<template>

<ejs-spreadsheet id="spreadsheet" ref="spreadsheet" :showRibbon="false" :allowResizing="false" :showFormulaBar="false" :allowOpen="false" :allowSave="false" :scrollSettings="scrollSettings" :created="created" :allowEditing="false" :selectionSettings="selectionSettings">

<e-sheets>

<e-sheet name="Registration Form" :rowCount="40" :colCount="30" :showGridLines="false">

<e-ranges>

<e-range :template="nameTextbox" address="C2"></e-range>

<e-range :template="dobTextbox" address="C3"></e-range>

<e-range :template="genderRadiobutton" address="C4"></e-range>

<e-range :template="dropdownlist" address="C5"></e-range>

<e-range :template="multiselect" address="C6"></e-range>

<e-range :template="mobileTextbox" address="C7"></e-range>

<e-range :template="emailTextbox" address="C8"></e-range>

<e-range :template="addressTextbox" address="C9"></e-range>

<e-range :template="addButton" address="C11"></e-range>

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-ranges>
<e-rows>
<e-row height=55>
<e-cells>
<e-cell index=1 value="Interview Registration Form"></e-cell>
</e-cells>
</e-row>
<e-row height=55>
<e-cells>
<e-cell index=1 value="Name:"></e-cell>
</e-cells>
</e-row>
<e-row height=45>
<e-cells>
<e-cell index=1 value="Date of Birth:"></e-cell>
</e-cells>
</e-row>
<e-row height=45>
<e-cells>
<e-cell index=1 value="Gender:"></e-cell>
</e-cells>
</e-row>
<e-row height=45>
<e-cells>
<e-cell index=1 value="Year of Experience:"></e-cell>
</e-cells>
</e-row>
<e-row height=45>
<e-cells>
<e-cell index=1 value="Areas of Interest:"></e-cell>
</e-cells>
</e-row>
<e-row height=45>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-cells>
<e-cell index=1 value="Mobile Number:"></e-cell>
</e-cells>
</e-row>
<e-row height=45>
<e-cells>
<e-cell index=1 value="Email:"></e-cell>
</e-cells>
</e-row>
<e-row height=82>
<e-cells>
<e-cell index=1 value="Address:"></e-cell>
</e-cells>
</e-row>
</e-rows>
<e-columns>
<e-column index=1 :width="190"></e-column>
<e-column :width="350"></e-column>
</e-columns>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import nameTextboxTemplate from "./name-textbox.vue";
import dobTextboxTemplate from "./dob-textbox.vue";
import genderRadioTemplate from "./gender-radiobutton.vue";
import dropdownlistTemplate from "./dropdownlist.vue";
import multiselectTemplate from "./multiselect.vue";
import mobileTextboxTemplate  from "./mobile-textbox.vue";
import emailTextboxTemplate  from "./email-textbox.vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import addressTextboxTemplate  from "./address-textbox.vue";
import addButtonTemplate  from "./add-button.vue";
Vue.use(SpreadsheetPlugin);
export default {
name: 'app',
data () {
return {
scrollSettings: { isFinite: true },
selectionSettings: { mode: 'None' },
nameTextbox: function() {
return { template: nameTextboxTemplate }
},
dobTextbox: function() {
return { template: dobTextboxTemplate }
},
genderRadiobutton: function() {
return { template: genderRadioTemplate }
},
dropdownlist: function() {
return { template: dropdownlistTemplate }
},
multiselect: function() {
return { template: multiselectTemplate }
},
mobileTextbox: function() {
return { template: mobileTextboxTemplate }
},
emailTextbox: function() {
return { template: emailTextboxTemplate }
},
addressTextbox: function() {
return { template: addressTextboxTemplate }
},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
addButton: function() {
return { template: addButtonTemplate }
}
}
},
methods: {
created: function() {
var spreadsheet = this.$refs.spreadsheet;
// Applies format to specified range
spreadsheet.cellFormat({ fontWeight: 'bold' }, 'B2:B9');
spreadsheet.cellFormat({ fontSize: '12pt', fontWeight: 'bold', textAlign: 'center', verticalAlign: 'middle', textDecoration: 'underline' }, 'B1');
// Merges B1 and C1 cells
spreadsheet.merge('B1:C1');
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-spreadsheet/styles/material.css";
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
` -->
```

Sample link: Cell template

Copyright © 2001 -2024 Syncfusion Inc.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Worksheet
- Rows and columns

Illustrations in Vue Spreadsheet component

Illustrations help you to insert an image, shapes, and graphic objects in the Essential JS 2 spreadsheet.

Image

Adding images to a spreadsheet can enhance the visual appeal and help to convey information more clearly.

* The default value for the `allowImage` property is `true`.

Insert Image

You can insert the image by using one of the following ways,

- Selecting the Insert tab in the Ribbon toolbar, and then choose the Image tab.
- Use the `insertImage()` method programmatically.

The available parameters in `insertImage()` method are,

| Parameter | Type | Description |

|-----|------|----|

| images | `ImageModel` | Specifies the options to insert image in spreadsheet. |

| range(optional) | `string` | Specifies the range in spreadsheet. |

The available arguments in `ImageModel` are:

- src: Specifies the image source.
- id: Specifies the image element id.
- height: Specifies the height of the image.
- width: Specifies the width of the image.
- top: Specifies the top position of the image.
- left: Specifies the left side of the image.

* In a spreadsheet, you can add many types of image files, including IMAGE, JPG, PNG, GIF, and JPEG files.

Delete Image

- If you want to delete the image, just select the image, and then press the Delete key.
- Use the `deleteImage()` method programmatically.

The available parameters in `deleteImage()` method are,

Copyright © 2001 -2024 Syncfusion Inc.

| Parameter | Type | Description |

|-----|------|----|

| id | string | Specifies the id of the image element to be deleted. |

| range(optional) | string | Specifies the range in spreadsheet. |

Image Customization

Image feature allows you to view and insert an image in a spreadsheet, and you can change the height and width of the image by resizing and moving it to another position.

Height and Width

- You can change the height and width of the image by resizing.
- Use the height and width property in the insertImage() method programmatically.

Top and Left

- You can change the position of the image by drag and drop.
- Use the top and left property in the insertImage() method programmatically.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false" :showRibbon="false">
   <e-sheets>
          <e-sheet name="Employee Deatils" :showGridLines="false"
selectedRange="B2">
             <e-rows>
                        <e-row :index=1 :height=30>
                            <e-cells>
                                <e-cell :index=1 value="   Mark"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=2 :height=40>
                            <e-cells>
                                <e-cell :index=2 value="Id"
style="bottomStyle"></e-cell>
                                <e-cell :index=3 value=": 1001"
style="bottomStyle"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=3>
                            <e-cells>
                                <e-cell :index=1 :image="image1"></e-cell>
                                <e-cell :index=2 value="Gender"></e-cell>
                                <e-cell :index=3 value=": Male"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=4>
                            <e-cells>
                                <e-cell :index=2 value="Contact
Preference"></e-cell>
```

```
                                <e-cell :index=3 value=": Email"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=5>
                            <e-cells>
                                <e-cell :index=2 value="Email"></e-cell>
                                <e-cell :index=3 value=":
mark@gmail.com"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=6>
                            <e-cells>
                               <e-cell :index=2 value="Date of Birth"></e-
cell>
                               <e-cell :index=3 value=": Jan 3, 1985"></e-
cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=7>
                            <e-cells>
                               <e-cell :index=2 value="Department"></e-cell>
                               <e-cell :index=3 value=": IT"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=8 :height=40>
                            <e-cells>
                                <e-cell :index=2 value="IsActive"
style="topStyle"></e-cell>
                                <e-cell :index=3 value=": True"
style="topStyle"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=10 :height=30>
                            <e-cells>
                                <e-cell :index=1 value="   Mary"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=11 :height=40>
                            <e-cells>
                                <e-cell :index=2 value="Id"
style="bottomStyle"></e-cell>
                                <e-cell :index=3 value=": 1002"
style="bottomStyle"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=12>
                            <e-cells>
                                <e-cell :index=1 :image="image2"></e-cell>
                                <e-cell :index=2 value="Gender"></e-cell>
                                <e-cell :index=3 value=": Female"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=13>
                            <e-cells>
                                <e-cell :index=2 value="Contact
Preference"></e-cell>
                                <e-cell :index=3 value=": Phone"></e-cell>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                            </e-cells>
                        </e-row>
                        <e-row :index=14>
                            <e-cells>
                                <e-cell :index=2 value="Email"></e-cell>
                                <e-cell :index=3 value=":
mary@gmail.com"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=15>
                            <e-cells>
                               <e-cell :index=2 value="Date of Birth"></e-
cell>
                               <e-cell :index=3 value=": Dec 8, 1989"></e-
cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=16>
                            <e-cells>
                               <e-cell :index=2 value="Department"></e-cell>
                               <e-cell :index=3 value=": HR"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row :index=17 :height=40>
                            <e-cells>
                                <e-cell :index=2 value="IsActive"
style="topStyle"></e-cell>
                                <e-cell :index=3 value=": True"
style="topStyle"></e-cell>
                            </e-cells>
                        </e-row>
                    </e-rows>
            <e-columns>
                <e-column :width=20></e-column>
                <e-column :width=280></e-column>
                <e-column :width=172></e-column>
                <e-column :width=160></e-column>
            </e-columns>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
        bottomStyle: { verticalAlign: 'bottom' },
        topStyle: { verticalAlign: 'top' },
        image1:
[{src:'
AGXRFWHRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAAD10RVh0QUxUVGFnAFBvcnRyYWl
0IG9mIGEgeW91bmcgYnVzaW5lc3NtYW4gc3RhbmRpbmcgY29tZm9ydGFibHkuIDt0SWEAAAAVdEV
YdENvcHlyaWdodABVbnNwZWNpZmllZIrxMzUAAABGaVRYdERlc2NyaXB0aW9uAAAAAABQb3J0cmF
pdCBvZiBhIHlvdW5nIGJ1c2luZXNzbWFuIHN0YW5kaW5nIGNvbWZvcnRhYmx5LiBae8yiAAAAGWl
UWHRDb3B5cmlnaHQAAAAAAFVuc3BlY2lmaWVkozTnUQAAs/pJREFUeNrsvQmQpOlZHj89Nvj89593Zlv
```

lVXVXX9MzrRlpDh1IYgUWEgYLIQKwWBlsrwPbBBActoNDDlgbFrBgl3vZ5XRgmzA2axtY1hAYWFh
sLiGB0aBbmqtn+qyuM+//PvZ93i97wBuxEev1tDQj/19HRWVXZf6ZWfk93/Oez2vVdY1mNatZL/5
lN3+CZjWrAWuzmtWsBqzNalYD1mY1q1kv0uU2f4JP7fX0b36/53Q3bdftD7L57XGdTOu6yq+Gfqv
Ksqi8/4u+p2z+Si+NZTXR4E+ddeN3f9TJ4tlZ2/L/xa1nPr4brWZekRa+I/ZTZ3Mz8NvtYOP8FdR
pliYn11N+9k4QrqzaXsrD/60XhD9x5Yv+0Wnzl2zA2qx7sI7f+1P90/3r/2AVR29bLeajfLXstwZ
b/eF4DyfXb2L/1jVMp3N0hwN0B114nQ78wMH44kX4pY3KqtDpbaBMlmkazeZFli9c13/c84OvuPy
2b180f+EGrM36L1zP/Mq3+5ODm78gQHxDlRXj1SrGbJEgSQqURY3xpbN45HVvxOTGTaR5ghvPPAv
HLQhKAe4WnP4QvW4H5/ZGsgkq2K6HOM2RphlqeXyymh31h72n272tL7zyBd920vzFG7A26z9zPfl
/fJs1Pd7/ljt3Dr/q4OadywhbsCoLVZZjvspwPJnK9wie28alhy7gjW/+bDz34cfR2drD/PZN5AJ
IOxC7uCwwny+xtbOB+y6fQ7c/QmlZyOMYlewJ129hemcfYa911B/tfjRP48975B3fkzafQAPWZv1
/WL/6rs9/3XKR/Yskdy/Np7OgSEukAjrba6PIMyzTWpgxRlEUiLISLdvG5UfP43M++y344Ac+gjp
PkUZTdMMQSZYhXixRi0PbG/i4eH6M7TO7iISdHa+FoqzQG23ptWoL8MPOTRTJPxGW/a7mk2jA2qz
/l/Wz73yzs4qc33Zd+9VpVvVn0yXKvEKrE2Jre4y6KiFYFQCuxJQtcXwyw+3DY7hejcUyxuWHz+P
tX/BWvP/xj+HGU1fx0COX4ZQebty4gc1egGWWYOPMCA9dPouWMKoYxfCCEGVZoi0+bibfa6sWEh+
kTl09LgB+84Nf+D9kzSfTgLVZf27983e+bSNKkt+x0XpUPzDHghAdOq1Q/uMijieaLa8EfFkWIct
rrOYzZEUptwtcP5hhKWbyX/j0y3j0wdfjTz/6BK5efRpvfu1DuH59hm7HRTQ5hNXy8fDDZ7Ax2hW
TeoVKGHWZxHJtHzvnL2B5eiSmsSes66PX613tdDfedukt3/JE8wk1YG2WrN/47ne8qXTb/9J1vXO
2fEyOE8AKPAQCmCw6QiQ+al7FSFYF8rzEbDEV0OZYJSUqYd5FnOHwZI7YcmBXFb70i98goOzjPX/
yFF5+6QzmixzT4yOxbifCyEC35+CNb3wdotNbqMUMzsXEzqoAbuBjICA+Pb6Bnd09pGIX2/nyaLy
99y2Pftn3/UzzSTVg/a96/f6Pfc13L+eLdzphN+i0Q4zGbThuIIxXCpEKuwoo48WRgBOoUaKQj3E
+XWCVZiiiCPNZiuduHWjwKa5cJBUQRxN8+X/3NnzoY0fI6gyj4TaefvI5rGYnaMkhMEsiPHLlLB5
77DHsP/GnqFxXyLuN2XyCs1ceRLpYCdF6AuoRJsd3EPrlfLx97p2v+/L/9aebT6wB63916/Gf/Wb
r9Oj2b1ZZ+sagvxGMds4IyuYCVB/wWji69SyS5QRB0AI/uzQuUMSx+JSu3I6ETRMsl0v5f4V2u4f
9wykmywROKOarsOp4s8L5vZfhieuHeOChl+FP/uMz+PDHPojz4y2sqgKjfoDP/YuvQ1BUuPHkR+C
0AuRegN3ds7AFvHFSi3+8gN8aYnrrKoabg/lgY/Odb/6af9IA9h4v5zu/8zubv8KLZP3JP/1G6/p
zz747mq7elJalm2dCm0WK5WKO2dEhbj37NCIxXSdCkzeu38bt/TlOZ2L6+i0NAFXwIPawMJ6Pk5M
INw+PBNAONns9TKIaTwrTHp8s8MjDl5AsIhQtB+PRNj760edwspyJqSzXqCv4dYHzF86LmR0hShh
HsnE4WSDojNAfbeLw6lUMtvfM6zo+CBzHftP+k++5cd9rPv9DzafYgPVTfn3sF7/Dmq1W7+4PN9/
AAgXbEmNXfEMCllHZsNVGuzWALX5rQaYU47fTG8D3Brhx+xDz0yUKR0wluU8rFN9W7pGmBSJxRrO
6RMdz5MvH8STGYrXEw5d2MTm4jb37ruD0dIUb+7fhCXNmRY5ep4Xx1kB8YxeTo4UCeB5VcOQQaMl
zzmen8IMQd/YPUcjh0O61AzjBG+e33/8rey9/U1NAcY9W03XzIlhXf/0HrWUUvxtl8YYsjQQkFnb
PbKLTBsoqQSyAu318KubsXMsDu+MORru76IUu7HKO+85uYHt7GxkrmOZLXNs/gtvvYbC1gXa3j8P
jBU6nc2R5jNGgjY88fVsAO8VqKX5vvMTFvT1sDjfUtE7tAPvHc1y/cwxXQO9128KgM3QGffGJhck
P72C4cw6LKcuJfUymJSyrhenJdGzVxa997Dd/uNV8ovdmNV03L4JV2PZ/GJ49+4ZBXgvYTpELKFe
nM2G1OY4EpFkOHMxmwmBbKCoBXOAJyASYrO0VJnSE+cJ2C47jIYlW4lsCp2K2dgdDVGL2joQNT6c
zJGJSb2xsCVtXeN/Hb+Khc7s4uX4V/eEluUaJtLIxGPZR2h5u3JniwkZfrulimZbIj4R5+2MBeCa
v10XFY95t4XgpprIcMEXuYLGcXW53B78tv/mM5lNtwPoptz76y9/zA3m8ehMZMRXgMapbxifIEwG
hfDqbww6Wkxn8fgdpKSZslGD/QFiPJqv4mHMBZy4PKsTw7Xe7GPXbYrYWcJjSWUwR9jpYLBZa3FD
GHm7eOcIwsIV9Z7i8tYl0dYozu1ewt3ceTzx7DflxgnO7Y8xOTwSkY/i+J8/V1dxunWeo5MuSbWO
VlvzOVTN7NV2ZqPE0wuaoeM37/vfv/qpP+2+/rQk4NWD91FlP/eb//Bbb9f9ukdEknQozRQIAH92
NbfmewfFWqKsputsdJGKWxqsVYmFF2w8wEXCvkhi2bSv7rZYpDg4WiIsCAwHsroC822vBrSwMh0M
8deMQm50QyyTA0XwuJq+H/ZMTnN25gOXJbQV9laXwwhb2j5Z45ct2xFcOFMxJUSLodnB85wYuj8R
vRkt+JgcHzePeEGIQIFsl4sseI6/uD5K8+MF/92Pf7BeO81Nf/LXf3/TLNmB9aa8//flvf3h1fPV
fdnauBEHbx8WHHhV7WEzRpZi9YuIKLDEa9bG1uyM/S4SxNgQMEzVvZ2Ia91q+mMhTAW8qzFpgazB
Cu5VjvlwiK2sczlJMFymGAlzmYUetLk5WC7Q8F6e1gzitce1gCd+6hQuDECd39vWgCD0B8+EpPO+
CmMUZVlktLFphNV9hvmJHjyPm+amY3V2w0yfPc5zI74b9DaHeQF4j00hFfzWf/NCzdybf+sQ3fPE
zD1y58IVf8vU/2rTb/ReuJhr8SVi/+1Nf+Vo7q3/r8GQynh3egtseIIummC8mwlqp+JoD+I6DZHa
CZHKIaLYP2xG/NLQx6ngYDgJ44p+2hQXFWRQz1dNC/rq2kGZiEPsOytwS01iWmMpTYdJQQGrJNRd
xJKCS31cFFvMMZ8Ybwt65XK+FWbJCkWVotTrY3RbGzOR6qTD2/iFy8Y9dJ0Cv18aJHBbiImO2WOL
a9ROM5PW6bil+rC/XdcSELzCbz93r12/1w1b70mjQ+8qnP/BbX3D74+/5N/c99tlFswMasL4k1u/
/2FffX1XVbyzixZl98RGjqMbR0QJpAiznOW7dmmD/1nWxeSyty12sxE8UxrpzbR+3j09wuD/BVMz
XduiiJ2Ztf9hG4ISohD9dgadd1/DF9E3zFB1h0zovxM9kF04OX+4TCRtmwsTMxdq2g9AuBYQVqqK
CttfUNvZ2t4TVu7j2zDPaqXN6MkEl4N7Z2YJlQ3zUCI7cEFzjudv7GPQ8ef1iC1eV+Nhy8KQRjk6
m4J0Hgz6mp6ed08Pppdr2v/raR949OHr2P/6HC6/4C81m+M9cTQXTJ3C97+e/c5CvZo+Xln355PZ
NDMZjpDOCKUYRz5QZmVNdzieYzGbY3D2H2ekcz159BqONTYibiTqFmL0pyqzUIJMtAEwFSLblCmE
6ci3IAZCIaRohl58lCVveXLgCTJuQthws4xh+K0SS5mg5OcbDFlrdEVj/kAoCd89dwP6xsOlygU4
Y4AYBKex56b5zKASQ85MVNrf7ODhNcfXmdVy+uIvldIKHHnoZ2p1AWfhATOmWvObjoyM5CFI8cN8

```
ZvPxVn47Do3nabvtPnjl/319649u/9qDZFY3P+qJbT/z691nx9PD3b9189nJ/cwe9jV1c+/AH8KB
s4HxmYUnAJhXKtMRCfNG2HyLav40zF3ew2X0M+zdvCZhqCBxRlZb4iAMNCNm2i1Zdivmai/9awPP
baA8DnB0PsVhk2myeMoLrOvJpO5gu5+gFHQGUhTpLxFzOBcRtOHmpihH3338Fd25ew/LoUIDaEb9
0Jc8hpFlE4rvWqOIV2nJoHO2f4Fj84/vPbsIu5YRwGC6Wg0cYdrHKxaxf4HSxUvB2O0N0e1uYTg6
QrZbBwY3lo888+fEPzWenX/EFf+sf/mqzOxpmfVGt9//Lb/6Fk+PFOyYnRwjE9/TaLraGFxDJBu5
tnUUmDLgSgJQCrFL8xCpfIRQzd7ZIMReW7ThMzzji16Yqu6Jtci57TwVkjlzPDeD4ngZ8LPFT7fL
PRB0y2qt2C5H4pKxgXEaxAL9CIvdl4f/O1gBbu2cUzIe3b2E2maIUBk7kMLAsSx6fYndrLOzbUsk
Ygl8cY1Se+MVViePTEwx3dmAlMfqDDpYZD4eOppPGoz4u3ncB8XKFqMiIeq0xptmOyppfeuD+/+m
Lv/aHv7fZIQ1YXxTrqX/3vW+ZTQ5+ZXY4CeaysdvdDbQ8W0xR8enGe0jiHFU6U6mVdHogJq4AsnI
xOThCvEpkW/vwB2NhNTEtHUuYtQQlC23PR9jqwGu11YdkUUQtLFemS/lgC60rLitbQFeothLb7LJ
KuFn8UgKplM+elb9yDqgZO4tibbXjSuU5srLS1JBl1ep7iq0tZnWq+V09BOS+sZwabhDIAVJqZNq
X18LKJ18sg9HWCP3+UNUrev1QAJuImW3hdHqCcBCi5bbR2xnMz++ee+dbv/L7mrxsA9ZP7rr+h/+
MluuTebS8TBMyWUxQJvRRJypeRiZMVwvxV+l92lp4QNMTAqoiIgu1Ndob2gIcq9D2uDAMEbYD9Vl
d1zdN4YxHCTOyha7U5nHhrqJQH5j443XJlLYwYikALARwhbA5Ga5MxGwVBuUhwZyqGNMCxFyDU7k
A1PUCNYV5/VzAmqtWk7xuAl0YfSGWAEsl/HZHbsfY2hjJ6+vADhx9PvrJq2iCC+f20GrLoWK38ey
1m4ijHIfinz/46MPz17/hL37da9/6t3+u2TGNz/pJW7c+/O6fWS6iyyhjBH4HlTBdt9Nj4FTLCeM
oUpEyIS8BBzQgxN+Refu9noCwFCZuiVkbaw7UcwkeH05gC6O1FUgsN2Qu1RL4ZOJb2gKO2q01guv
w/qWLWgDuC4hrDTPx921huVjYN0MRpnpoFL48L01mvyXAFp+2FKa2HTGJbQ1QVQXN2wClPGcSJpr
PrS0Pg2FPCyOsThcbNK2zSCPMy9lKbicYboyxMx5jKe/t9sExzo83cP+lHTmgOhgf3MLHP/ihvhv
4P3HzuY8c/OWv+cH/q9k1DbN+4ln117+7fe36jeeWR3fG6eREfb0zZx9ANDvE8Z0DbI66AlTxC4V
FZ7M5omWs5u3m5hgdMZHd2kKvI2auANelMew5FOUWJnU0YOSIz2q5tpq3tQCzyJbyVQpI1SPUfte
ayBcmtARURe1qKkY4kz8SgNUU/Jb/CZOmKwEs00Qucna002/2PO360VM9EB80TfSapfE3kYvZTWE
1MLIsQIwzMcHlIKGJTfZdLiPM5TDqy3sYbg1xZm8Xi2ksrL3ASMzqjhwO4XCE/dtTPPnMTXQGrZv
3PfTKh/7aN/0vq2b3NGD9hK4P/Nw7f2dysP8mT9AUJTU6smkFAQI8odA6w+RwH7bYqKuTifirgh4
BQ19MyF5godPvC0g9BAIEu1zKbd+AJhTzUkDiOMKO/F6L2VskmmXNk7kWQVCWlGas0K4wo6WFExC
wEsQl866VYeJS/FsW5cMKFISZmOcZwS3sWzKPY1WocgEeJdQErAxspdFC2dZiHphNW14bsZj3Jf3
egjKnXZRiky/kEGIqaraKsJBDIJAD58zuNrqjnviqvv4uXsjrhZjNAmQ7GOHq9UO8//EPveef/d7
1phGgAesnbl39tXf1ju7cfiadz8dVNhNwhIIjH/HkGO1Q/M2whXwWwRKztczFRBbTtSWs2ml54td
5mgf1qbvkhWAYyPaMmj59RHVQawd3PzurFACKuZnHwqxlJj6qQFfAXzi1grJe6wor0wrgSjFTa/q
thQFuRdwJSzMQxIBULWYv87j8R5CXVS6vdyA+bixgjbR2wg3bch+BsbyeQvzjWA4ayxfTWg6mWA6
MsNPG6dGR5n6PFgnmwrwsddwc9LBzflNeq/i/0VRNfAbGHDHbxztbuHOwSg9PJ5//TT/+W/+h2UW
Nz/oJWfOT/Z+38mR89NzTApIUy8kSq1mM8XgbZx97JZaHd9AR5ownU+xevh9uOkdbfMEgdOB1u8q
cHlnOZfBoAKc1MKDyffVvLdujbarsKDtf7i8gc9gNU8JhRFi+PDGNS0EWeZSAEBpFaSdyuxaTuBJ
T2NGgErWWFJAapBKflswppOnaofrYTiEHg7A3U7V14CroLddVU9sucn181+8jErBbYilUWYFITOk
N8VOjxRJdVxhcLjidzcRbdhFFN3Dm3I48R4DjwylaHQ9nzuxpwGl7qxecPbv9k3Lph5pd1DDrvWf
V3/gBJ4/nzz37+B+cW05OkYgpOT2YYDzs4uL9F9Huj8AQ7er60xjuPQA/LDWPykJ6v90T01L8Ukt
AJ0xlOT5smqlUjhCTlf6iTRkX+qQCZitfqknJvlbWBzNdQ5alymEhDMpVrc9kMjjEJy0LI9XCQgd
GhOWZNGpcMnpMVhVmpn/Kw6BYp2lo+rJsUV6UXEh8W8vV4gzuH/qoFGmr2m0dwbGKhYHl8Bhf2MN
8wmaDGofHU+SFhSiP2FGPwLdx9twenEoYt9fHaiXMLMx79uJ5jIab6eaZvTc9/NZv+KNmNzXMek9
XNDt6y2p2PPbEv/SCWMO8586IvxaUCJ0cgVMhPTrA5u4WOpsj+GWivaF+KD4mTWSCVUBBhrPJoAI
wi2kdYVn6pBodEqa0S6rl9xhJkvsJMxKADA4J00F8WccLFMjUP2RKx+M1GVxizEh8ZjlDFCB2wbQ
Q1uY1lR/kYBDWL3hfpmzkGjSbeV/mXCthZDK4K4dJHC01+uyIOVsvp3BrT3O++TLF6eEBRhvbiE+
PMJb3eXByir4TalEGK6+W85VGvct6hW6PcqcjeQ+++OsD1iwy7/pYs5v+bDWyLvdgZdHiXbZtB57
nqILDeHuEjS0Pg15HfFHxPdNCNuYQw/FZbXVrt30xf9vw2n0BbR920FaT1hZwsYJIA0U0g+Q2VQ5
rDTZVan5CGddRs5i/s/xQ/NuOmKm+YFZui8/p2CYgZckhYLtM+zhaRWQKKVytgKpdgtHX3/P5XAG
V5nBp7trEMOuPK30NLJTgl9ZUBb7Kk3pMKzmsomJreqVm93IRYSVsPN7agGeL9RC4WnG1MeoJs7K
4I8XB0cT4u5api66LFVJhe7/d3f3Ib/zEqNlNDbPes/Xsv/9pxy7TnXgxEVYrVXTMEdoKepvoCRM
xIuzI5nRtH63RBgLWEHmbsuEFaPQr5b41Uy0V86YCUbsnP69UfV9uqGlqkQbJusKOFX9cGO1gjdI
yIMV0it8RQmcRv6WC38zFsoSwZrRYQMm2OHEZ4eamY40pGoqwab5X7lt5jqodMsZEYCqb83SX56a
Amv6qYAGkPGerLWbzSg4R+V+aaRTbFVN7Oo2xcE9Rb7TV7N3c7CE/ZvljIn8H+Vu0Heyd29WDp78
xhO/JgdXpacnlycGd8WQ2fdfDwN9pdlUD1nuy4tkzF+J41WdhPX3Hdn8sTNdDtx0KiwqzVhHY/+K
K2Sj4lN9t6GApLbQnGMX3rO1Ka2stu6v6vxarmwhYYUgGhQibyqq1JJG3C5sSMPI71gwz+BO04Ah
QKwJMSwYzAVJLwFdoiodhCgKVdjCtZov+sbBtLf/nczHYdNccJiOT8UC9RLIfTWcq/NdkT0t9XHY
NYc38fB8uKyTA/tsWTk/nKpMaBLX21+6MO+LT+vpYX6yOdl8AGsiLkefsjzfFKhFmTS0tDNnZ2f7
cZkc1ZvA9W8Vq9l2O1+37QReV00VSeuKLDgSsPTF3ZWOKyel0QtmgbTjtDmRHKyNVjr8+Phnl7cu
mb5kcae0oE1peS6uHjMMJbY0jMMmoHv1SoToNFhJINtvlfDWNazV/qazvmWsLuGxeQ/xgW8xkzq4
hoyvQ5KsSELHmWM1mmt0Eqe3qbTK+s76v+r6WqjEJrkv1YT1hdlZU8XriAyAU05qjPhZRJP5pJAC
```

```
3MZsukOcpEmoSxymOnruNo2efRRJliJcn8LtdhD3xXeXAOTk+7PzTf/hXnGZXNcx6b8LrcF5Visk
6W1ISxcXuxh78OkZ7OFbTEGLmscDB0ZoCAQZV0dz2uvEbWoigPqTtqy+qaRSatRoBEojQIq4Lkz4
RgBAApZ2uMVyp1rD+XMND9GULFfpWOsW6MF9t50y7eO76oHXto3S0Onlt8lZqxtPcJn27lY2cT87
nsNkfK0zJHlq5D3O6Fa8hh46ViWluV4aRmUMua402V/I+CwFpSw6pilVXDFSJCR7HMQZnLooZPMZ
ikeL04ONaV1y5A7Q6nX6vJ3844E6zsxqwvuDr4HDaycs2jhcVdnY3ZUPO4fV7yoJax2tTgLtSc9e
2Q1YXKADtNWNy3KJdGXOSJqajBf68f2WYk+kVAasGnpRRy7WfSbNVwGPl8vtKywGpcghjNBuWpK8
qPnJtsyoKcJiuqY1PWmnlkmFma82ayqJ3o8laUWzr9Up9kPyPKR45RDTAxfOAuod+iMApdfxkp9P
CahZhFrF6sCNgl8c4K2FuV3O6rcEIVn+Ak9lCGPdUTXDmks92N9DvsZd2EYin/Tny4KbAvwHrC7t
+4wf+tpPk8NPkFJujbbRaQ6TzCcJOH36rRyNZWC3XTWm1XGVPCvDSH61L06PKnCqZy5i0tqntJd8
JK7P+l+CkWr8ifF02qIzOYv6y0hY5vd8a+mreKuBLYyorqgpToM/UD69NNmb6BbWaupx8rq+BeVz
HpGkUkLwi72tXpqeWbXrMs9KCv8vyws5OUWlnULRciPnf0TxzzMAT2FMbwfNN51BeThG0+3KI2Ei
FnalGwdrmyeQUeRYzuh1YRfWlDVgbsL7wwaUk77Q63SCPlgiENE8P9rG7t4uwK74pm8HddUTVdZ7
3Aa2aBQqe5lBpddJLrNW8tLSMUEFZrs1jPkZzq5YBEr8rq2o4WJjYMkxM01qY22JahiBms3euFb8
G+OWapWlea2Co5uVNhVR9d2NYyPj4dceNTkBnBFmB/+fMfprRAtSCQSjXvCf1l50UnrzPQg6Btvj
ohYCU1wjE9z44mQmIK4w3PRzvH8AXk5pCa0cnSy1JpO+73Ojg7O4Y7Za31+ysJsD0wv8xHZxfLia
BK+yyFP8riiL0ugNNgdhrs5FJS7OhTX5Tc6g0NS2TdhGHVoFk07wlSzL3yCiusmZhgk76ZLXW/LJ
h/e7HqPzJIgYn0OvVAoBSdcyqtS8qP5NDoyqNWLe5LtSn5aX+rJqtUnPc+LOuif7S12WBxvNvVo1
zE31e+718BfafZ3SyNNv0PBNV5vXZ0cPUzu3TFY5nS218ZzeS59uYnkxQ5rb49xu4c3CC2zfuaCC
qWQ1YX/CVxNmF46M74pi2cXR0gr5sOqvOtdhAfUzZuJZd6+anOUqQ8rtubJJalekUOJbzEaTq165
NX1SpmtC2sJOyNDtp6kQZVwNBPATWwNH/q3dZqI9LM5blhxVLCHk5KlGwNLGkqmEiJm2q4OXiYzW
9w/PDdtbXLjRXzCIKoxyhlf9m89QGqD57a8nCDosu2loWqRaCPI6NBaxVJoOn8hp6/RYG/RAH0wU
8ObSy1JQtnjt3DlmeYDmfYVfciP07xzg6nfjNzmrA+oKv+elszKojNm7HwniBW5lyPfqZtqkQIqM
SCIZtbTUzK6yzE/RlCV4GfKDyDgoiuy40r2mpnZyrr8qmcQPoDBZ7UqvcBJkqcxgY4JlAEKPJ/Dn
rhym/wgHLGoAqDKgLBnaoRCr/lwNE+2ANyKHBrPo/2TJkdGtdJPE80/NwcB31jTWAJH5AVcR6/yJ
LhNQDVVmkfjFLIvuczSN/m+sHhxhv9BFHKfIoRtBp6yyeQn7XH44xn2dZs7MasL7gq6iq+xhUms6
X6GgdLRQg6jfW1frPba+LHEwrmqZvbCPpolKhbA0XRlLz0w617UyL5+kH+i1TzWDbz4OeZqZ1NxN
ZmXwrqrVparkmUmyZ3lNNCVHL1PaeN3kry36+TU5NYdyNDGMNWE3w6GHBi2mk2VpHp0sTyGIQiqD
ma/Y45DlZahmjz9tsMKDbTb9WQMr3TOXDPEkUsAR2FC/FN22JRyDMXBrNi8ViqamdoN0MpWvAeg+
WbMQZS/BS2Xy65cQ3y6K1yiD9Q5qYhRb1CfgcZTj1O7nhafqS0Zjaod9KELH21w3MdzGtKwKYhRG
2EejWVjlhrHoNYGU75kUZWbY82CxNpL/phNoIoIUQTNuoLKmn320lX19fx11GVfeTZrpxTRW8bKk
r1oGpqqiNOqEcBlrPyy/9uZjUaarBKVfYlJIytgavasOulaOPCcM2stKY7qH46RPx7VlHzZJHpn4
mpzNc3z/FMk5QpA2xNtHgewFWx741n840tTKdTrGadNENhRHphzJnCWPimiKEWjtZbMtam6u+Aq1
em8a2KqC1FLR2Fa6LIBjgSU0+NaVZXRqfkiDSgn9X/U2NImuAKYUTdlHGkXlOdt0QVJRqYcG+RpU
zTb84HpUkHBXxrtTnJUqhLI+cdcXrPK4WZJj64bK+2wS0NoVpwjNwxuCTU2pQqSVmrTVJkDKnFdZ
rcx3aXEAwZnmNQDBMFmb7nJ1FmuaZrMQ/LxPs7D7YbXZWA9YX3kwp8VxaIs3zNGCDd8m8I3OUjqv
stLZGhYGoUWiZnlQ+0PGf75xR8DKaK0xKlQYW9GvFD/1a9VVjUxRBM5i6vgw++fJ7YWayaEUE8VA
QwFU8ICoT0a2Dgfi2Ky2GYIcNkVgyGMQGcwGIVXkajLLVbi5M5JlPSb9WLYJSI81Wve7S0dk6plB
CSw/hPD8WkoeVrdfytHKK0V/LRKy0qqpi0E1eayZnSkvM3/lypjKmPLAOpqcIggB+nOFkkglwo87
1D7/Hu/DIG/LGDG7WC7YsL7xNBlsIY5A5okWEnMJjdW6KDZhCYWECZVg0mks27KhiAs3DWm7DH8I
K+oAnt70WyrBv/FYmbtmf6rXhBF35WVfVI2q5DW0EaGmxvsXbNKXlWioDSi1hBnXYytbqow578r0
D12lpAwH0MX01lWu237FVzg31MIBGq01Aqa5sDVSZwJcJOqmKIq2BtSmv2k61Ke7gZAAFOn12z0J
HQJkKAEcDeX9yClBbqpD7ruTvM6fekzyqFXAYdAnfC7QRn4x9cnLSn995atzsrgasL+hq9YZTL/D
TQPxBFu1HSYH9/X0tatCILLtWCFABhio1WKZ8jxHYgr6nMq9rNH8d46/asnEpDapsK76e7GS5LYD
0uqqbpL2rBLGYzAwoMXhEUFcsA2R/Kx1Qb/18ch0W79dMrXQobOZpfTCft+TcHCfQQcmmespjttS
kmdy1P0ybl4dMYYor7qaJrPVBdDd1RD/XrdiWV2tFkqVWvoe0FNOWiVXf1TY5pnqWSSpsb6lg+Hy
1NK13YppTS2ogJvRyFgVHt69eanZXYwa/oKvdHSXR6WGqKQzxOaN5pBU7VBD01Ox1jMmrABBGY2u
Ylu/5Kjtak01VBeL/ceF12Z+CiPchKGhK07xWhmYetFJJF/qfJpZbaSdPrSauq2Y3/dWa/a6MBiv
rwUR2eWbzNTGqy5RQsbY42XYnZrXNEZHiH9vymIq1yRrBNi9Sre676R4+N0FbZdrzqqY4AWoz/1r
r6A2+Xt8KtNWuqii96qNsC6OK+TyZxFrlxJpkzqj1WraYyT0c33mmqWJqmPWFXX/lW39YCMpdMcx
Dxfoki5Em4utVJkp7t2pJI71iouZppvUO7EopqJSvOVf5Kl2T9yQA1hVMFRXyEZhraA1/ZdQHFZY
MVhkgVsKcPBCokaQdpPQbyZLaeSM/810Fa8lZqowui3laesYEJvNCTGUVYmP8uLQ0VcMvvm52AFX
rLcOih1Ib4kMDeEtfsI720EOFOePS5HzL3IyTpD+eyTnAie25uAKuMHacVGpZTGaJMG+lxM2JBLG
8NdYRMy874/jIZjXM+kIv2d9LyqssV5GqA3JjctZMO/A0faKtZILQbDZR4bPCq5ThsjxCcXjT6Ck
p+9YaTaXMClMsjgDD6Y7UF1T/WBvASwVtncZwvJbpXaXaYGm6emqX/T2FppHyxUxV+jMxL5nvzMX
UTgUMeVabiLCYxnkiZmiSaA2xoy1tGVJ5HC0FT9NAlprkWom1zsfWMM3xrFwq6kKB7ZbQSiT10Vl
wIddhADrk4KzSjPQotY3PUiZl4Gn/dImNXkeLPhI5xOaLBL3toQ5zDhzrVrOzGrC+8H9Q2/43wq6
PpFkVZMul2MaB1r6WTtdULMmmnd58WnyyWoM/QoeYnk6wiiNQDCJOBVCUWmEUVcxECooNN4fYOHd
```

```
FcHIH7e0LApy2ydlqGsVSv1RZNqEJG6npyRRJVVH4O1ZRs3g5U7kUDp7KxFy+feOqHCicIperEr8
tjNsK5HBZztUE3djsY9Traamj4wtHL1bqs/YGfTHt28bkxZ8pH6qKPxvrs8oo9pP0eXCoNrExlwO
PFoS8TDko5quVStxQ54k+albwdeVotzrCqLGyLv3ZsmyneVmdNjurAesLv/zWP4WX/H3B0phiYrF
sfDIrgy40dVdH11XVj3NtCvHd5tMj7XNtt8Za+RTFc51gzjEafleYJVmpYj4DMpUvPu7JbYSjXfj
KpH+usJ7+ITt4yIDMt1pmHCPByRGSy7jQ6qWy9BCtSsS5MClLF90KSTJDtqrl+QVAcjjcd+khnZR
+49YhFskcw56HXuBpoCk5OUW3m8mB0VoPxjKdPZZVaaqm9MT0zep1wYZtqpZqUxvshj3MxY+P5VD
JK0+HPqsvJodNnifP+7xU+GcLXSLvubacVN7DfrOxGrC+4Ct0wsmyxkI28ZjscCpMpWMr5HeLySG
OnnkCvd2Xo7Oxh97WrjFp2QXD1IrY0OlqidnhHcwPbynj+Ny0nHMjgLNdMQ9nc5UntYYbykrMoVZ
2qfpJKrhtJVo2yMcybZSkC6TCZE7oCmN6qswwW8zVHLWCLjx7hd2dswhabYzPnFXVRZfF94spZqd
n9LCh8He8OMViOZXnPxWwJ2jJ8w+6woyBKbSo67W2E31ciqrJz1k66NqZFlTkBOFypXW/mRxePqP
aBY3pVJjZFeujXOs7iWUhr7cdCNN7Pfl5kgqYl83OasD6gq+/8T/+s/qHvvrzDwQKl92ixqA/guW
x0dtFdLSPzb2XYXj5FVoo4Lgsqre1ZJBVS2QWP2yhNbqI3Ys7SJYLM91NzOksilFGU/l9V1vcNPh
Ev5UmpoC01GYBqKYv1fMJIA6bUnEJMSspE5pWC238Ho83EXS68MXs7Q4vot8bod1tmz5bWraUZgl
6GLQ98VlXynDV8LyAfld1k2Ynx+LzRlitYjV/KQZXr6uYLJWBoW6TvK8404FauZj1acUG9Fi1jAl
KlhhHnIHj+8LiprmAPiy1m8SV0MfXJmi1+qof/Hdls7MasN6T1e8M/17ghr99uH+t3+n0NPdJP3K
0e1EYaUtM3MIEh8RcZN5TRRy0n5QpENMWlwsgbaeEK8Bz2mxB68iVe2paOr4pbmf7ndYQsTopj7V
ggaka+pnaKM70iPikpfiopvHbR6vlaI1xd7PWua9UGGTFkEv1B0dFZUxXDRUsQg+B1Rar3hQ8hCw
NdAv02jtYRQtMJ8da96t1xgwAr3PJlgoxulrCqJIxAthETN+VfPUZsRZAp3IgJGLiO+2hmMXJ8yM
lQ6+vY0RcFV+z0PJbTXFwA9Z7tzqDrT9tT2/Oq43Nvh8GWqzAQVRVRbHt0ohl26YUjyMu2DJGny6
lnEmSGzYVPzW+c0vM4plKg/Y3B+hvbIrZ2UXN8j0BsSWmoq15U1dLEAl+9VWTCBmVGTjJPDN+YVW
aua9Ka/QTBZgc9pgsQ2HZjhYwOL4DThHQFjoeApp+LdUvzeNIgz4dRrAJ/HAkwPd1viwDVzTJTWu
eAN5nWWGtLXE8XJhfzVI5gNajOFbim1MBkQOt2LyeC3B5X0a/3drGuVaASE6asCfs3u82keAGrPd
u/fXv/NHyd777K95rxYt3ZIGwl99WFUNOKFc1Q8voG1XCeNnpIRYHh8JiY1ib54Egw/GtG3jm2rM
4vLUPh72fgp3ek/tyCDi4+Oij6GohhK0leXVoChq0cohqgvMj5LNjZLmrQErEFF0tZwIMMqCjoDt
4dh+hsGN/74r4ziMxT12NImdHEyC+CZuHRbwQ0AXoDQMdhux0B/KcpmiBBRYWSyoJLgF4LeytGsI
qg2haAFHm60ixGYLFx5Xynr115FiLNKp6XTNtyUuz0BO/OU8T3Lc9MLIy/d5cyPzvNzuqAes9XZc
2Bl9hZ+M33YzzMUdo2BxJ4VqmNJDleeK/pbefxOEzt9G98hosV1M8+0e/jj/6wNP4WDRE99Kr8Ht
/+D6Mtvq4b2sXrwtiPCIG4fU//mOcf+1rhAF74sPONcBTCwvb0SmqxQnKkzsoxa9M40rMyxTz2Uq
uzZGRFpbTE9y+Psfe2UtoDzq4cXwNUZThg8/cQNEb48Yz78PZURcPjhw8sjuEn/s4eWKCKnkCFx5
9EMOtLe080FpgP9TUjK9T7tQmV23imlpNDHWzp1ZY1nOMIJvvMohUmWl39F+zGC3HU7Zm/tWxCGR
LGwOYmw1DF5fP78z3o/yJZjc1YL2na3dn/E2R0+nPDm6paUm1B21hE/PTUpWWBU6evYrMbePd//Z
nMUkdXHrT2/HoF34hkoMFWhsPIPnd9+Kpa9fwps/7Uoycqficc5RPvc+My2DXjTAzlhPTCBAtUS0
XKOX/+ZKC2bH4lSWmSzE/5YBYUt+o8LC50RHfcopcrv/6t34ZThfCbrcjPP3sBL/zS7+Mhx57BIv
wIp74k8fxuu0Ozo1HsAfn8MxHn8bOmWPsnD+vHTYlfWqqOoh5b6W1qirSBC7K4vkItapW1EYnzhW
WThZHAsptVfY3LYK18XPFzO91Oyx+wBk5BEKfg7kEsMkqe9X99xfNbmrAes9W/Yf/2gqRf/mZM/c
F+9M7poMEsWk2Fz+tXC2wuvpRLE4j3L79NAZ+F3ubI1w6fxbhA6/CH/zzX1K1v0cfeQjnL34uqni
F85f7GNohjm/sYPax5zC+chk2e8zWA5MhZis46Dir1dQtFgvxXYVdBbC5tULOpvjpKUt/0X7tG/C
Kz3kreruckbrCwfs/KuYnh2Vt4+yVV2Jw7iGczXaxOLmK937gTzDqOdjptDE9mcITP7k7HGnkmC5
ona+M/KnWE5suHR23sVaiYM2zJ+zJlgFPm+rF0m+1VD2CE+FZK8w2917oYStoYVPAT62nthui2D/
tDax6Qy5z0uyqBqz3ZlVVR+y8HuxUWEXYxRd/lW1g6VzAFSA/uQW/NcLlN96P+xiUSUskwqz1ZIb
sA7+Nv7FX4kZ0B488tCEMfICe+KGDp2pE02NsjPZw5uFPE58xkksRHB0d/VhqY7mZxVpbRta0EnO
YxbiZIJRDkY+eu4MxpwEsVlhc/7gGfzqdHQzyCNMqwtd9y7fi9p1jJNcex5U9MdfTmUqkTk6XSIW
pz9Vd+ELrtTx3Z2sEry9+rLB1qRVatja9c54s576atrpCvzi73eNsG46XpLiaTZnTWr47aNmVgNf
HdqeFvW4LroC3x7Y++ZtU01X/aHH4d64A39Vsqgas92RZjttm+1vgpGgHoUZaFUlUOKKiys55BP6
mdtuULLAXbgmFJRkdLecR9vwWztQJiihAMSnli7q7S3gPPITelcfEHBaWWh6aNjnVHA4FvAttmav
rUxXgZo8GGSqkmLcANClKPPKZr9fKJ7/OsHzfH6O+/qze542Vi0e8GLOrH8ejVYXWZg5b/F+nb2M
YbAsYB0Y2xnfgkA3ltVariH0BRuBN3oKXZsjEnrDqdSE/c75sqJfDyrfMNL1xv68MSx83cBz0A08
Y1EFP/O77BbADOQh63VDnxZYpyyfTQEyML0QD1gas92wVmW0VVeCENs6c24NNKROXozBCo7jPQch
agkeY5qYtzue4R/Hf+iHaoSVmrGx+29Ts2htb8kFtQlxgOOUKXjHXyHLtr8XTtGF0XZRwtx+VQRs
vRztNBWAOfNcUzYejHjrDDQEx87VyoDBAJOzbFnbd7sVGloWQytmqNpQzpmMmq8v9syzV92JVqSp
XOCn908KknlhVte4UstnOp5FfT8zmEC1vgaEwr6slEo4Gkgj8jXaAUF7XhXaIoVejGzjyesQjFrc
hWcnhJN9Dz2lkXRqw3ksz2MosH2mwsYlhbywbPINFGRQ2m1umWqeuY73NSK6ClHpMqqxfaleLMnA
QGrkUFjjYLlpuIaw407yn3e0JILt6LYv9oeXCPDV9R3vdrdPKhfE89ScDYa9c7pNNbiORawfDLR1
5UdRyH5edcT3UdldTMERdhZ5KxOhgKblfkkSwc04DKLXDx+EoyXXOVqfkFCbHSzO3LO/+GSrN1YZ
i8sZyfnCq3DLT3nNhVF8jzxd3N9EqU4xoAut1jcB5Ke9B5V2ymddsKLOaftZ7EWDqbyb15hiV10c
oYAVb0GCUAnV2DTdzyRY31yhICEiNSn7xvDQpy/JYSOH12mgPehj2LRWKcCjvwtGNLORflxxWqm/
mmpRK7awlQistVAgF1GHoax6z7dnYsDL46TEQHavsqRY+6PS6zEye055UaN1uWa2b0hnNdRzV+aW
```

```
/qa17jmWmBFBreA3O2qrWsqrluuFcnl/Yv+u10Rf2bMkb4MHRYhWW3Gtva4jVnZsqQWqH4Xouj7x
uYVSPJjfLKuWa6a/+pNXsqoZZ7w1Yu4PUSkfisPbUv3NL0/9pfmm0g9Vslc0I3ey+So6xCF6lVLh
hhW2prsCRjSij9YhIo85ghS2Vf9FRkMjMVDot9y9N2aKCzNOcrsfp6HIY+BV1ewtlc/qsYTaFE8W
qy6QzWi0YUTZWQrEdll0yLFqQnwXaNysvQwBU8EaeG61g1wh2QxUL5fWKOfH8SCzL1r53pmzcVqC
Ra1oOvl2i3Wlr3+u1a0eojo9RrirkWzF6/S46HqVsxE3o+KhuzeH35fFnrzSbqgHrPQowtQdVvQg
ye61gqCM0WAhhrwdS6fBh0yCu3zk5nDkVFuaTFfN0Le9ptj7TJHWZGClSVXpwjL5vbTR562ypEVi
L+RCVJHXMFHRCTcxfr/JNs7eYrSxS4H04R8crXTiLU1UxpAgbm+ZtMT0VospqUC1g1vZyTo+tc1o
FtDSp5fWWYr46qlZYa9634DR2birKyfC1UZzcrxC2WqjkeejTjkIPg3YXWVpjtljCF3OcJnayXMG
X1xW25E4C7tZohOVqAe/MEM5eA9YGrPcKrHuvqK2DD+UoVmpSejqkyTaCZESGbWmlD2rZ1PGpafC
mlKhOcqtVzVAVIHScRq36vwRyxfSIqt/LtfJSFSeoGUxd4loAzhk2qiCjahSpajdVrCryxcSkJm+
5nla+1ksCr+GLWUtyXM1NdZLcl8qMLHioa1O7az+valip/jFHZtjMHWtt87qs8G7Em/50YJsxHcL
YvJLnrRAGnuoBt+VnoZgJ3XZbyzCz6QTFItVIOWVVtcaZuk1i7VOC1d3u5di5WDe7qgHrvYkv3X7
SEmbTCgCXwSUV2A5UhZ9kSb0jzmVFlaw1gwtTkcQmMc9Rv5MOrlq+9HFtI/9p66/Wc3DE57XySmV
OHQL57shGKikSzL4nQDKT4jjFjRPlwKYCmtvskmEUuvZUTI1yMGqkKyt7erCYca6Vmb7O+l0VKHe
M6hN9VduMxOADXctFvm421yZ4ywjGqaMp74cjMDgcmeoWfstBR4AoZIqOuAmOWABJvJLXWiBzU/W
rzeDoCp4wePu+B5uumwas9xKtJQtiff5562qlCvVkI1U1dGqjhE+/TNFoK0Ppz2rL+K62u07FGLP
WUj8yVZDquAuLvmJm9JrkvkUcoyzkKd223CdWRqOCtpE+rXQQsxuyf7ZnxLfZzE69RYKUDMmCfM/
Rulwdtqxtbn K/wgxPpiYUu4KIPmt9Pb40PRA4FItRqlqPGgWxCr6sB0LTD/ZaHvp5ICbuQPxRH11
5eaeFi4yHkJeoaFs+P9XHVME6lqTM7MI7/+DvNRuqAeu9M4OzyNLJa6WtKhAsflB2Za2sHWiVkUV
T2HM0CKQTzBmQUcFSGF/v7rxU9XHFHxVLsU5XQmqsEqIZ6pgJF/RxyaZ3O1h8Yd6kUjlPSrpQaJv
pF6ozhK1a2MxMbyNjm5mrtoqw8f6aC7bN0CyrSLV5nKMzFHReV4CfmRpgdg4xWMX0DQXEYQS/qU5
Bi8EOWipVylY9erE6sKrTEZ+0Da/bhy/XDpMC6WSuqaLuqIs8i4wU6nooVhFFCHutI+/s5e9odlQ
D1nsXDS5K26LagjVdTzh3dU6MynlyYBTNTHafUFq0Xk8fV9PXMekQYD2MuFqPrjBqEWRYDq3SqiH
VATbsyYIl1TyiKqGYkDRzHQGNaxlgJ3miQ5fTYiFg6KK3MRQT09EiBU3dWMY0pknOtAwfo9FfRod
toxVc8WeZEffWQBhb4taT47QvlyYzudU183oqmuL6fkqNcOt1VUFCDi45FOYntxHNJnJQEPiuNie
4muItTZ6VusH9zqL9GW9v6oIbsN5DkziZV9p+zWqgOjWsynrdu0OcdECra+a1lqY7hWWHhCALCZy
1VjDqlWkW52Ap2pcCJlvV9WHm2IShADmWH1vawG2tVR5YQlirGLcZgcE2NkaiWcEUzxba6O35Y7j
tttYUmcCQPH+rszZlLZ1NQ1+6QKJAtel/B20dMUkLnwcFI9f8XqszvbYkmDJaP29pmQILBqvYu2q
3Qo1CHx0nWE1Zb7wy8284KY9DgrIU/U5f/27V7BS9l539V81uasB6b1eRV3UWA1sXUTGXyT+zZab
Dqc/HWl7LsI5hXjE/aQpTEpTTyXVkZKwRYd34jKkSpIGjZX+aRvEEOGEbViI+sTCmK2ZkkVLiRUB
H97KwVL4FHLUYBhr5pYJwJqCJaX7q6A5hyvYAYWBm2+g4SUZi81wA2hYLINYm8NrONbdqq2w4hb4
ZJfbU32agS4cmq7iEq2M7NIqsw7EYdXa0HNIiy2uTQYn5wSkqYfte6KLXCbGkwHe0QLcXaBEHLWE
5fm4OXvWZ/6jZTA1Y7+0f9TWfX9W/9kMro4JPVs30T20rg63FslXdYR1IKVP1R2sCbzET/3JlJq1
1emKudpWB+cU5rVSOKMUndDojVbzH2sS2MwF4tBR2bAmjZardyy4YT56TqZrKNXnPqhRGo+L9nKx
W6AHhDLbhsTVNCy3kkWRZFlPQDHbNvDudoUwTHpmpdaZZTPJn+SG1fzl2gwEym+LdvlEUZpCpMvN
ptceVZYwC+u2xg5EcNq1WgO54A7ndxnL/CIG4AOzMKQ/voDfqPd7/6/+giQQ3YP0E+K2tUUb20/L
Z2mgTwb77567VfLU0QCSgFBMWq4myZDE/Rh4L83SF8eyBMYc9YaewtX6kY/xEnfgGY4qKD1gIeAi
aomyhptxLGRsf1DYawhaDT57Rd6qXEfIoEmALQ0e8m4Aw8IxfyfQOWbMiK6cmkERWlwOFvuvdemA
zSb3Wg6Is1kOW+ZwVB2eF5iDCOji1nr5DQ79YyiOzHJ3NLoLWAFZ3KM8bwJuLRbEQ6+DgUADrXB0
/9qp3NLuoAesnBqxb53/PquJHGQ0uBUisFqo1FwKNqDIXqkOLWXifiskrpm8+OUI6vYmq3UHQ34E
dCqvmpvCBYy0YPWYqxGYDQDTRa7CKqIqXGpwqWp5GX4tsgXytwQttfIfp8pHHe2lXzMwT/T8lR0v
flqefqR/LgcYaCCPA89hMXrWcdVpmPWjKWk+N4/zWNRgZvGLJoSZhqclEg1sVEo3/69QUdMtRimu
QLacop2JBJKH42Zb6u27XV3+7PD2A59ZpZzz+kc1v/Mm82UUNWD8hyy7K/76upl8qt8ZaqM8aXu2
QWZcIEnTMnbIbR/y1YnqI5OAZFJ0Wwt2L8Hp9IeKWkQdlzXCcIT9ZKWt6g76qJLJgnxVE1HMqbA+
FFSKKpohXKx2GtTo90aomipr5/aEKm62OD1S5wm+ZaK3DSgoBWrI61WBTZ/uippMcYdNSyxmZvsn
0UKFEaqF+aLWOpJU6V4emdGkmaZlgFS2JIjVlizraMlUrIpmttA9W5zXzfSfy90g46uMUbRZNXHk
QrXT5+JUf/uUfbXZQA9ZP2LJe/dYF3v+rvytO4Ds0/WLyMWaoU22qeyoOfhEftTg5RLR/FVkVo33
+1Qh7mxqV5SRyaz2e0aXf63ng/JxMy/FCBWuZJsgpBi5gyZIFksUCq+kE0WSpA2Y4prEWXzeZ3oL
JzghAxQ9lr6mqE9q2KYZg4CmawZ3dQntwDqZ9vXy+7a3SAgtaA6let6jNYcF5O1owwcCZagfTdF7
PmnXMa2eNiAbWqIgoB4NjW1qGaKXy+OUMCDvCrh46l+9/Ynuj/+Zm9zRg/cSbwjv3/VX79uI52dD
nTOGvmXrOIFPNulzWB6cRsukNRLMDtF/xBgRhX4Gqpqjq/BZrMTJoesUdCANxYltp6gdyAYBWEZH
RhH0duR2KX2m3QxQc20Ef1a4FnK4mYxnpJfOxgMKkeGoNErGJHC0f0eIYQSCvITCC4qW22wlHOpX
pEKJvyshSYUoZy8JMjjOlS8XzZrGWMMpzOuJrM+1UMg0F06Prsl5Ynlcn4xX6PwzPXbi5+zlvf7N
9+dVNUKkB6yfBFD7zcFnMb06wuH5O84xM+HOza6BmPQpxeozVwT7cnctoiQnqaBucKYTXYK+2vlR
i4lqqh6QupBtq3tOpfDNA2fc0SOUJQHLHVr3fenGCIlrJ7VLzq/QN2WZXeZZeA6q9b2rzK4psK7g
yDUrNTm5iNL6oEV+Hmr/CljSJWcGkQ5qzlY7rUF/aNk3RbM8rtRLLTBtgugl+W8DZkdckZnch/rc
1gLU8RsgRkXIgVGwY6G1h1mvNnYdf+7lnL7/6TrNrGrB+0lbl+v+nbOZHuaHZskYz0tJ+0wB2TF/
```

```
1SPzMCDuXHxM266jL56zVIly6gGLC0pzUfCvNSgZ7KIDG4U9ppE3rlhbUV5pjpQpKLuY0MmHX3NZ
ifHbRwDHF/axwuqtEyABXqWAPxLSW68ap5lzzZKKyMb3N86b0kAGlfKX3NTNZmb6xVGWClVe19ub
WKjF6t5+VaSO/1RVzvQ+3PUApvvPRcx8U9zfGwTLFz/3hb8l7qmCJSb473nrq57722xt94Aasn9z
l+L0fEsx9tWXVfZbnGYkWX/3RIksQH95E96FPh7dxdt2YnpKwtORPSw4JLgKGYymK2EioyB2cWIA
yOUQ2O0Khw5etdYF/CWc1h8cIsyfmbpIqfZasLGJBvxuqD6t5WNYO81yglj67a+xAQFVq6mZ+cgf
t3obWGlOJnw3wHLrMActkeaZgjX6LY3RdmOJhAI30zQg2fWq2BbqtdSdOgVx84lJM8eu3D3Ayj8X
clgOh38H4zAW/2SkNWD/5gSY/PCjK8th26j4tWkdnqpr2TEKQE8/HV14p9wsUDKzyoe+nTels3mk
FGkV22PsqjOgmppoInLfqD4GehTLNtAywFNMzIFOKH+gGG6gCAWWQKAuXdqXBKAWpE2gtL3Fm5RR
zETBxqh0nzgnr+46pPFrOb6O/c86kgKxyPaC5UlO9XI9S5mvGegKeFkXRZEeuYyMtLY2s9F+2SHS
zMWA1EQb3fVeHKfdaHQw3Bg1YG7C+CPzWnUdq+/ofrbTvUxjGWqc32OFC0LpdMRUD4c3kSCuAIL4
czVZ1+xgtFlBW8oV4ohVI1nKh6Rg7rrUogv4v6ZHlepn4qA7Z0lnP0nESLR8sCaY2Cwd9VYpgg7u
nmiuO9sA62kkTGPCqOlIKx20jXUzg7Ih/7NTrmmA28BWmaorMz9Y/Y9xrt07JCiZ2AFlGjYLN9lo
fzVLH+bGavJwOV8hBxGZ0Ff8OfHmJjeh+A9YXC2Dr6tfqsnyU5Xq1YxQObbutPa2tUV9F0OKDJ7C
azDC49BrNXRYHApTUN8CmacruFVf83YxRWduoEGqRP9XwbWR3YuSOsCdzpJYAIq8FFI6KZXNwchl
VOsWtsllXLD8P2MvKlrqWFlpUppNWDgALXneEQF5nfHCEUiVSHa20KjVfXGtJMCuZSqZ9GG1mdFs
VHqo1kG019bX9T8m21vSSbXraMRcLoS0Wg+/5+rNWf7uJADdgfZGYwrX9A8I0XyFkM6Zagsq6UBi
sNYTf62mNLpvCs6M7OFh+EPWqgCfbtzs6Q09VUzJlsoJRIHNVo0kL+WmSCtNl8Ryl/DgVE9trd1G
FXRSJSZWUXTbvLDW9wrJAFjfk8wiFb0YuVrZcKxADuiVmtl8jaIfae8rGAuZjdeodI9REGSPa60H
Hxd0mBOo2sWuIgafKdA3ZLIrw26p2oe2B4osXeayHEl8CUzauV2nwTCx97F081zSYN2B9cay6tXl
qpceLEuXYUVUFSyVOOFZDxdAKYc7BDvoXPRx95Cr++Jf+EMvaw3G6wETA+tfe9nnYee1jCFem6N8
dnkMtgLNYvne8QLmaa4G81x5R8AlFukSaLNj8o89R5gnSNFWTlB07TOVE8QpJnuF0PkGWZRhQ8rT
rYOfKGQSttpjZcxVbU7mZWhMz6+aD9cBk9qZaplywVta1jLoF3VvxVa2wQykJY46zk4idOL0OqpU
cDnIwDP1trYRKk9nRycnJtza7pAHriyMi/NhfrvP3/uMDB85lqAIKJVky2bwjOMKCZXQHTncXvjD
S1kM+Xv+XHHz4PR9HlBb40+eewrVOgMuvfwOCOEU1OYSzfQ7e1WuoOqGA6VXIrj6B6onnUAsj5r7
4f4GN9CDCcj5FIOyc1Svkk5UWKgT3X0Idh5i//2O4eTrDnzx7HdfjJT7rvvvxsp0hXv7pW8/7tZ3
tcyZtIz4yga8i5X5o0kf0UdeiT9RiqjV95Imf7KuiPvOrWsko7zNPI6MDRYZu9TEYDnEkbvL2oI+
ds4/97je96yeXzS5pwPriYdcSf6+2qt+W3d83KRozC8YZXTSFDbLJ3VAYdmuIvTe+Cq2NHl6/CvG
WZ2/jaL5CuRCzdnMbmZjM7a0zSFZH8v+LcNt92HtbCDbGYvKK77h3RqVc/I99CP2rz8F5+BXCpjH
if/8HcC6dh//gK7B84sM4fe/jKPpd3IiWgr+O5mHvf+wSws2esj5zr35/Q86UCHlipqdrjznTNqx
kWk8xrwjidYGVBaN2YYV9FSG3tBDERr6Ya68sI94QtmYU2EaA7e3u7z76mke+tNkdDVhfbKbwnyI
9ngsK+ty0zLcy2GP1NwVMifqB2gDeEp9QNvbGIw8h/+BTGLzyZbgsjGr95m+j2N5AuS2m7s6DSMX
kbQ/GRs6lNUJ+7gwsAbGrQ8cT5NvbOHjmw7hw+VHYJ7dwbMfY3T2HZHKKw499GBMxQVeLKT7z7B5
eeWkPF15+FuMrOypoxqHIZEItN5wvdVwk28kp4k26rEoj5EafWaPETAixSELHgNCvDk1JYmXeEwN
qXrcFdAZ6jd3zezxQfuUbfuRffXGzMxqwvuhW8Jp3lOm7f/wZx6rOsXtFixy40dvCQIVrCud1k1c
6pyagZtr5lZi9S7h+B1YqQHjmNqwPPonszlSjtvmn0awMtbChEGC13JYOaa4L5kFbmJfyLMcnSD7
8AbRvCEPu/yay1QL91QoPCAu6/QH6D17C5uUx/G6oqRSaAASp5bW1GydP5siU+R1VntBRG3z1fM5
aM0MoMnkMix8oO0P2pMi40zI+LvWjVuJ5t3roDncwizNceuDKv/iir/mOL292RQPWFy+7wvmbYkJ
+sPasfq06RaUKfZPJVEObA6VKS8vwWKzvXbwgDHSdEvioerZqLnni53oHU7QXS5QfehZZR1jw3Hl
VIywHT8ISsxbLGZznbuLSM9eQfvAnxReuMBA/MxdTmjnRmqr4Ax+t7b6Y1G0V5aaO0/PRa5ZrCJv
mSayFFCz2t7XHNdcKqVonnxemfnndZKC6SewUEoa1OVyZOVlh1+Tkthw+Lux2G3VviMXhR99nhd2
/2eyGBqwv6hV+5tdcy/7gx6/Jjn7U8n0TIaUig9dRitIIsWX8WUtNZGG382dh3zw2RfwbsuHJsHL
fKhC2nc+ExFy4UaXpnfrgOip24UQz1IncZzBAQcEHjmDkVA7EtFDRE+Z2+iH8LfEdKT9Kk5eMyco
oYWQ2j6MSJmW0WfzVSpXe7LWWkouc0jLstsnXBRI0hClFyoZ3NpsLu9I8pmJhIq/Jbnlm8p3vX90
5f/9nfNaX/d1GYb8B64t/VZb1E3aZ/UiVVoGl5qKjzMlCfXbZ2OvADYpEfteFvTnSAn4cnJpRGa5
R12frmQLSE3OYmknDkQZ+qISIO5mRM/VrLfYvOJeGLB46ci648He7yFiCyEBSZRhV5WPW+sYshGQ
5YiGPzZny0WFXRqFQ/VQxe7Xpnf9niaLfUsa2nbYZ1dHqy20P2eIEabaE43R4IKR+b/A3PusL/lZ
TAPH/czUjHz/Rq7P9j4vSeq6oUtMPyolstTGBHRX/No3aOkFcTFvtGN8SIN5/Xr53tAXNctedMx6
L92tlZEZzrTzWbhdGmSynMr3ubIGT5/HEjO6fHyI8t2nEtIVq2f4ml4DHIBCF3Sj3T1EzBSRZVb7
qdfFDbQ6RUkxifhVysnBkRsE6Y6s2SorUceRISvlFthJfd3GkQmzsuAmGO+99+We9/T3NBmjA+tI
xhV/1DnEB3XeVeZWWRaJCZDqMSQlubQYzR6ItcSzmN2kTdMUk3hrCum8L9aaYmWPxVUcCypGPui3
gKlbad8qJco7gpQ7N4/0zLXRefh7BhTOwaVb76wlv8sUZsB7HeAhrmhBXZUTGswz5aqVdQTR/c2X
VSospqE+c0z/mqB7mWlXQey0K7tHU7QmYYy2HLBanSOdzeVxy1N+99Lbm02/M4Jfc6r/p635u/ns
/9vVuUb2BDd1sfaPmEaVTCBysmUz1jzT14WuxPmuJ0WaN8AD1sCVg6olDGpneUmEvqxyo4j7SBB5
/nvdheZZqIdGXVTYW05aFRn6rZcCq0iuxaXmT3+uwKWF9FjLkearmLlUtaDKbemPzRYUllhIWJoy
tTfLt/lhTNVXOxoKZnEMr5IslNh5+9fvu/6y3R80n34D1JbnEtHxzXlUf86rqsiWggJfq5mfAhsE
mBphM8UGuMqI6/KkyI8bZj1qpFrCnpX0c11iHXTVjjeIglfoHKqRNMDNCzOJ9h6xLm5uMK0Cl2mL
NkY0+fdRCxchtqleUFFBbIk1zZPJ/BoqMGGKmLXSFTouU5wq4fSxj/rYGWr2U0WdmS9zJLUSHRwg
```

```
c5yi8cOGvNp94Ywa/ZFfvjV+flZb1rUWFlD6dKjCU5uy06vLPQM3bZaq+qPqz/NCEOTmf1cpqI0j
Gbh5tbXPVJOW11HzOBajxStvaHLnNPnKmQb2QNb7lGvCcPWVSLdSGYoQ3ExYmMHMGkuQFZmIap+L
35jo9XQDMlM1aL7jUCejC/BytkbO0UNg0WSCenKKUL28wuPPqL/nGWfOJN2B9af/xg94v1rCfY0F
QzTEXbBC3tR7IBI0IVqoFMtBUZFr1pKWKjLgKM2owqTASoJSCMXCVL52vmum4C8vKdXYO3eDKl/v
Y8kVzmvNrqH5fZcbk5s8AFQHP41Snzilg08xEkzmOQw4CttkVlRnGXHEciN1WEXCKRRCkLMqID25
iceMGwsEo7Vx44H9rPukGrC99dv1vvqKubPfbZcenOn6CudY0NeqAZl6FKeUrVgLmFaokEjDIbQq
H63yZXL98t9KgsV2TTeV+2UL90Dqbm5GTiMT1zeR+1rrpXUNJWtDAFE29VsyvGOUlk9J3zQjaTGt
/S+3UyZRhMx3g7JiGdsekm0oqXghI83iJeDbF0VNPo0uRtO7GvHf54Z9uPunGZ/2UWK3+uV9cLfe
/16+tyxTTdjjTpjbN3VBB8HIt+xkJSH3SsWCb4yxijdpyfIYWNLiGnet4TooTMzlSMJrp6YX2meo
BICydM4da+2bWK1nSMbNgqfpfaMom1S8GmNRVhplAlwtY89pDRp0nKhPmleozuRT0Ztuc/P7o6Sf
Rk9t+u43wwqWFv7l32nzKDVg/JZb3qi+p8Qc/k3MMhg51okqh7WtwSEdFgtHaREv9tAc8Z5wpWo9
YFBCSidnMHqWGdcnAZOgyU1Y2DJqZQJCOliRTWpqW4ZQ6V/xNNodrKaGYy3lqel+ZoiF7UhyNvm2
eE+C2ionrNHXkZjCWPH3OTFPtYnrjDoK4Qm97DGe0gdaFK/B7/eZDbszgT53lu15WWJ6mTTTQxII
IxwSLWCPMyema0kljWOIX2qmYt0VkBi2zjG9dkKAaTUFLtZyoMKEFEamAsLCNKUvFBmFU9qYWZbJ
uKjcAzuTOsQLVpG9iYfNUvhdq+jK4ZKkZrL2rRam+bJpE+j1Pcty+eg1OEmNja1NzucG5y3C6o71
bt259WfMJN8z6KbOYT7WdjoAhUgCFNfOWuQ6Foi6v5XdNhVEmTBauixqqUkHsVutKJ5jIcF3QzzW
TxzVCbJnAEf3STBg2zUotxs8t1kvJASBApDxoIT9LolwHGscCOoJUh10RmMq6tgKVVUtlacS8KQw
u0MZs/wRd8Xc3drZVUcLZ3JSvXazyNLDC9nfK0//r5lNuwPqpsWo78+zSFBlkBQrPYpeZdrFw6LI
2qbLTxcq15pYlSgSsQyG00pi6zMVC856ULPW1uB4MBrHAniCjaJkAOSuKtYgoDFtaZlZNlprobxI
Ju7IIguDntXPzHPRXCyMyrNKmHCXJ+68Op+jJix2NBur/Vv0+yvEZlPI990JcOHP+0o2bT/+V8+c
e+IXmg27A+pJfTK2UrgdfTN9IWCyRr7YOSS5VeLsWlnXDIRy/NOWAlrWenM7SwJgFusrClcWg0VI
HWBHMuXO3hrfWEY1xlqsUSw4zyoMMzLwqgUjRxFWSGV+V8qZ2rdFhgpURYz5OGZ2HBa+dpJjeOcK
408Wo19PUDwv6i/4AwfZ5JG4be2fO06wP4mj5LnmbDVgbn/Wlv8QMzRgk4qCogLlWAVYkDEt5Fg0
SeS2dhM5a3Mpbi5CVJiBUrYdDkfXKcqWKDbkOmKw1rZLT75R/SbVOsWA90KqokCSJHg6RgHhBAbX
EFEMw2suCiEx+lwlbZqzL0Ppfy4B3FWOxf4AzwwFGG0Ntf4stIBqNMHj1m+D0d7C9dxGe62O+XMC
13Eu39p/+ouaTbpj1pe+zZulMFfmFHSmMzfa4LGbvaVv80lzN0YIqg6X8rLKNlArb4eibJpGWJNJ
wLcU0rdzcjFMVwGuKhtVHwoTMnSYcUMVywsooHFZkXbkff1foOI9M00ZkXlRGSdgyYwQAHh5yvWg
6hZ/kOL85RtjrCOPWmHMw1v2PYucVr0UlJvrOuQvaIHAynWA42sRyuWQ74PfLC/6V5tNuwPqSXsK
l7xM4fLZlewF9z1D+Q9VfyoXaOvqNBQtT7XVNOAiK6oLEk84+TU1UN/2/2XsTKNvO6jxwn3PuUPN
c9WZNSMgSqJkkJIEGJDBgY7Ad2+nEjtOdwUk67nYn3e6O3Z1e9ortFa/EdjdZnSwbD01jQIgYCWF
LQkggRgmBxBAQAiSB9MaaXs13PkP/37f//Z9zn9zdiWPj9+S6WqWqd+vWvWf497/3/va3v911Bud
8qjOqgcs54Q37ntvbhVZwv+e8qU6U6w8UNGJtFd00WZ/PZ6KDPSiABgG3qM7hU+hrzTs96WzvyszsZ
YmCwdPcCump57j92RURl5ze2y5Iy16z7jwLGLpFEbka2NdZmeX5C2M+yRBiYERJecPPnMO44evXz
fYPeN9QL2rLXoq6QOJqnU4FmdgTSjPWmnkWxtrsg4BLMR7sLgOGpRSzMwMBp7X9XyAQShPxaG20X
Tuft3z4WtPRAaXNg7gMvNYvdcQY+IEhHApYxT7fwMHkydqwPdgrD4Hj3sYK8tdfc+Lzl6SBojTfa
z7rnPWB9fkKO3vE1mD13CSXiHL7nIHX9dVtdXZGFhgZRF5FljYxOQRW2OTjT3veu+sV7gAFMyfgp
KDGAbsctGEqk3R6SRtqXtDG53b8sZbxLqsCAtpH7IFOiBGRUchHIvDG/dP7pAceERC62PKpqbMJQ
FGQIgM5t4ojqjXM7ZSTOKeKdQh8C0OBeKp84zLkxNyOzCPP8GzKYzeU3Sy2+Uq268ReojU9Lp7Mr
Ryy5nWH52bU1mIJnKDh2RWZfH7u5uC1Qxtjbbl/zLX/3f/sdf/Oe/8pv7d33fWC/Mx8jkRtFe6zl
/2QS/L46dAUWTUncecyztyV6rJVsg+oOBRJZRSjBoYKUbWHmU+NmpqlCIJ6nkgHzUWWVfB3FQnBv
tdpyqHns1w8LlvuASo2kA4xo7LnTe2ZLp0XFZuPiYy52bnLu67sLqztz3ycyVr5JLLn+p89B9abs
N4fClV9Drb7h8dn5h0eXbPXKn4F13dnZJm6y7Y//iY48177rrw7/2R3901/YTX3ny9/Zv/L6xXnC
PuDayhdIIZFkgjo2eVlALayMjtCeEyNvOWJETUofJhbxod4s5KIp6hSzN4JFSmaVgGxtkW2KX3DZ
yTI2LCT4BIk4jwlFSOC+c9NCv2nGeu0bD39nYFsxfPHrxxTI6Me28ciotZ/DtuUslPnyVXHrRpTK
7uOTC2h0ypo5dcowh8O7etizOLblj7HO48twMgKUWc+N6vSmf+8zn5F2/9/t4bbMeJ//nK1/xstN
f+eqT9+3f/X1jvaAexchoB1qiFBHE/yDlENXJ20X/6Oj0pDPcSNrbLdnb2pReq8tBxiA+JL45PXZ
/h7INx80AgIpEEeJcVR5qIEikXfc1kJqz6LpvwYME6SBH+LpBPaeLjh2Wqel5ajrt9TLZm1yS8Zf
dyHGR8wePyMhYUzbPbsvEzIxMT09Ly3lOzMxZmD8gbWecSaMuUzPTsr21Ta+eOIP+xCcekve85z2
y5TwvlKayetF0h/qR66595bVffPwrX9lfAf+R2IYJYe0//vIeg6c/O5ae+fpzSVRbjDAlDmqBtRp
b5nKIoEGWFHlpvyM9hL+7W9Le3ZHO3q70Kb8CYKmnYBGKq84SGB4DHUZgDFUH56ohjoaRk7FXfRi
0erJ8epn/PnrskDOyOYknpmSQNuRsx4Xgr7hVJo5dIWtnN2Tp4EEyqs6unZXDR464QxyR1dV1lmi
mxidkc2dbRkdHZWxiVNbX16glVXOG+sjnPiO/+7u/K9vbuwzBGSlgqFVSx+vXFpbmrv7UJx9Z318
F+8Z6QTzS579UH3z3se8kEh3lTFNoCkNUm5hPVxJ2xghb0NDbSlTYeckUjeH4GV0y3ZaCS/Cg6Hc
F4Z6Dm1N6XOgvISRFt83O+oasn1qGfoRcdPFhmXFesjE+yybyVecR+wdfIQeuuVF2yW4qZMblnl3
nNdvdthw4cpi1142NLZkEc6lWl01nnBg21XTHvbJyhhEBvO3jj31e7rzzTllb3eZYS21ScNGA24h
qids86omMjIw8euDQgTc89OCn9iVK9431AvCsJ7+eFM989rlI8qPssuHsDJ1KDm+ZNEYpqMa+VEq
LghLY53gLTnhDY3inw17VnGiwB55AwnfGmbW7zhNvsjF888yKM/ZUDlx6iUzOOQMDIypusnyzVkx
```

```
I4xW3ydzSpXJ6ZdkZVV0mXUi7ublNr7iweIA9rlubmzI7t0Dj21g/K4sHllgGOnnihPOWTZe/tuW
xzz8id999t5w9u6klJeeV6y42j2MlzYH1hC945vHx0Y9c/tLLf+yDd969PwJ931jP70e+9mzU/8a
Dz8SSX8Z5NxgORT3gSBvIm6MSj0wwdCQiDCX83IW90B7u9dhEzlILSjWDVD0rlB76bem78LSLsHl
9Vbrdvow545ucX5Jms85wVIq69KQpq5PfJ7NXvVZGJ6fkzKnTMjE1ztB5e2NTxqammZ/u7e1Jp9O
S6dk56XW6BJa0ntqXkyePy5gLg9ec1/7CY5+Tu+66S3Z3W/oZ4pvb4VXpUZuS+HAYxopjmZgYf/9
bf/AHfvqXf+nX8v0VsW+s528YfPqbUfqth56Ja43LEFZChBtEfRgdbg9GZ9Qm5t1zTZZBUCBFLlo
MupRvAWsp7+45I24R1AEpApPSe6096bu8lvNqID86MqaDkqnNFJE73Jq9QvpLr5DxQxcxTN5yOen
U/Cx/f3Z1jchvs9l0Ye8GDQt56e72Do97enaGBry+tiL1xoisLJ+Wxx57jB6111dWFHjMRLyh2kj
PKvTIdci+NGuav8bOYEcSF1ZPvOvRzz/xD/dXxD4afP7ehMPfVwy++SB4RaK9FRDxzRVUUrdERYg
IItqYW+N+D20lYd8rJpP3qKTP3RciZvBmNp280ZT6OLxyzE47GAae740flM3Jl8rY0StlZmJK1jY
3ZG93V+YOHGBYvbO1K4eOXcT3PHPmDD1rw+WkZ8+elXFn8KPjYzTg1t6OjI6NyYkTp5xHfUQ+9ME
PSRvMqrzghlPDdACpa7N7riEwIgSgWqQ8uvOFNhQkYvb2Wv/glptv3P70Zx79n/dXxb5nPS8fj/9
f/2LkqiMTx2ujo4uYjwpeLhhNQIMzEulHODc1Hp1gwzlEk6Jc5Usxi7Xo7kjW2aQkS0Eyfqqza1q
qBlFrwLgznVbuct/0otfJdv2ATM8vStIckfX1dY7yGBkdl1arxfddnF8g6QLeEmR8vM+my1Xn0Vj
uNozl5WUaPb5OHH9OPu9y1Lvu+jBpjcaisgnpeA2MloOWCx3PEXlvi6FYsX8fbAbIeWfmZn7xU59
85Nf3V8a+Zz3vHjtPf/XObPGVi7VGXafF2S+gdO+8HEZjAKKJ3WKHtHANrXIIZslUGmi4jLGMzkh
6nEzOCarC9/PDrCJn7IO5K6S78HIZOONfOnCYRrO6vEIvCbBn8+yGM54apVm6Lifd2dmSOWfQvV5
HWrttWVpaoiGe8EASqI4nT56WRx/9rHzorntcTqzgFoy0gfNIlCEFWiPKSzg0hMFJUlPyFKRN3fO
o71J1giMlacz/8vbbbhr5xMOf/eX91bFvrOfN4+l7/2D+2e+evPElLz0gS+OTUi9UhZBeC2R7yJH
2B+xEzaIuJ4sX0A6GFKifkUoPzJGRLuwFCQJ27kLNOMLIjZpkjXEpLn+T7GTOQzfHZG52lnRFkBR
m5+eI1gLVBY83adZlc22TjQFTUzPOYHcIIAHxhdddPX1GRicnXMjadv/uyJcef0I+8pF7GaqjRJx
lXpkCEqh5AmUosrEa7rhdHEADh8HD46JzhyGxKE+5n/e5aeDcV1bWf+n617764tdc95q/++/+7e/
uh3/7xnoegEubZx5o99LF9TMrsnDwYuf1GlKHgdabKteC0Tf1Gr2oMn8HJOeLywOh04RyDbWGgQR
T4iVi7ROGQNrh3EsluvS1srPblcmZKWmOjsiuy01hrAhp8X3XedCDhw8RAFo+uSqNkURGGxOytbF
N8j4QXxj26uqqTIyNM28FovuFx56QO+58r3Q6HW4UFnKPJE0CWgzTUbaBAiqMN9IG+ziJw++hBwX
kG1PXYah4D2wO/Lts8F8/+rlHZ2+6+fof/+xnHvsrX9bZV4r4S3ycfuz+l288/8wVhw4dkme+/Yx
kUC4s0jBekfpLQFDxcw6VhrYzzC5JDvRGqMPmAyVGZMoRJrBUr0s2dZFkV71D6lfdLp1+JNOLi0S
CAQrhMTc3J51Wl+WcxaXDbGdbObMsUzOjLnccoUGCWgjiA4x0Z2sbBAZZWVslIvypT39a7vjAe8i
YCuqKbBzISfC3kBYIMr7MY+IY0eyO3yH0rfnaK9r58DeZL9xod9HAefa9Hz55/ORXrr325XP7xrr
/+Et7tNdOvj/Ku1PI3x7/5mlpb60SiS1yP3qx0DCYnThwsQCdBl0aLaiHRactmcsni4Gfmg7ixMy
iFJfcLPWr3yLx7FEKoc0vLvBvz5w6KePj4zQ2kBWQ087NzbhwdtcZ8SoNuCgSWTu7Sq+Lks1zzz2
nYJE7jLW1NZmYGJP773/AGer7WGdFux68YJIUzHeRqzJfZWkopQFiI8B5wTgVRBrla9j47sdJMlb
G5iSQnNG/QwShrKz0Zf1edvrG177qv9k31v3H9/yx/syXf2J39dRLm+Nj8h0Xen57dVe2V1ck39s
moaHgVPKM3pXDjP3YC/S7srba2pRBe8d52g7RX04nX7pc5Mp3SO3Q1dL3t3Zyeoph6sb6JoEjhKs
wVBjm2NgYObv4/cyM87Sdnvv3thxcWqSRfPfZ79C48Xt42omJCfnYA5+Ue++9V3NRT2yAt0zzwut
AZdQhVsMcoVe13BSbD9lWCHOLnN4WxkuvS6+c8gvn3Kw1CU6hjtsYBQDWaHZ6vX933bWv/OTNN91
w1X7Ouv/4nj3OHv/Or4zU4ua2Cxu/8dwZObE5kGXn+WYPHZWxxqS4mJMlHDaexhF7TgH+MM+DQj/
Ap7xPIbVofF6SS26QZOFyerdOq+1C2FEaAXJN5KUTU5M0OoSWi4vzNAiUYvCaJBnjzyMjY/wdQmX
mp+5vzm5uUMh7fHxS7r3vPnngox9176ceVQn5LpTF/FhsKB4JpmoiwtnMgCSVolEwSeNcM9zgNeI
aX6dk/5hyp5BFReiNTQUGDq1k92m39uP+l296/fUPj42N/sTHHvzk3r6x7j/+wh7bqyff4QzqkjH
nmdrdTL67sStb/VxOnNqQiy5dl9rkgox0GpJORFLHOMccmktQF3RhcNyQCCWZqM/wOFp6iWSHrpW
4PkEFCISmE5MTDKNPnTrFhY6wc3tzi4Zs5Zft7bMyPT3JEPX0aRf2Liww7EUI3Nrdk6kppR1CxiV
yXx/8wB/Jo4982r2+O6TfxOHLkXrSei3hd/HRABQYaZjIp1FRSvshH0ckEIt14SQ0dGjAEYBqJG6
DUBnVpfmZoJARu00BkjTdQbs5Ukve6jaMU6+78bp3O+//Tx586JMvesR4Pwz+S3gM+vm/GpubbUL
yc7tfyIbLPTGU/LmVLdlcPyuD3W0KnEHqgTNrfG8qWPwQTSO8NOa87yWvl8HR10t9bIJ9r1jUCFX
hSU+dOa1dMUlM0gMMcXZ6jsaML5RpAA7Boy4tLRCNXT5zhlrBMHCQHmCoeO0f/P675VOf+STD5MR
3y5DTW2+o10QLn6jhUpANVMNBLxAjKLMaN90xjEqzoeT9GmrGudZfIegGoj9CYkYEaGZwNr+4MKe
emo2+ddWh8mUqtt3n+ZSLFH5uZ2fnqdtvu/nWfc+6//hzfexsb77ZGcAl03MHZO3ZWHZ6BZFeMAx
PbLRlfWVNZheWnTcbYddLOjouOlQqUvS31pB8/qUiM0fJbBpxxrLnjHvEeU+EmTA+1ENBD+y6kLf
lcr7Z2XmlCrqQFmQGeM2d3V0Kes/Pz9IgAR7BuGFoq6v4eVyWT6/IH/7hHfLNbz/FPJozcUB8gIZ
xTvyX59RMfAgrGhYjdzUjo7YTyfwDHW6V5hVCf51UxIh8Z/dc5Of2uL8HpREZQIahW17cLXXvUSe
qXHcRtpJCMFbWeeYrnZd94I233/x1d54/fP9HP35q31j3H//5Fzxp/Ea9XmuiINMrEqK8oOLVk0h
WdnNZWd+RJReyxlM7EjtDrYNaCJAUPa4LV7uw9ypJI0Vd8UDNdHJymoaxsrJCI4Ax7u4AeOrLwYN
L9KDL7ncgQ8BotzZ3nXHXCR4BODq7vsqe1nZbO2kWFxfJUrr/gY/JU9/+lhTOy6McA8+IEgs4yWj
XKyKNPEnYj9R4YajIOeE58Ry+s+wUqZfFcGd63rRg2QleFAwm1G1rdUWRMeQZ70PD9wqOaMBv4D0
LfIrbFOqWL0dsnnef0uyn6WvcLvKNt7z5ti+7XPct93zk/t5+GLz/+DM9Wq3d6wdp76UjzkhcYia
tnW0XSqpXGnEGu9PJZGVjj/klumXS1rZ209RnRF76NkmOXSv9gg3bzBvhQeENEX6CWQSACOWYra0
```

d55Ujll8Quq6trcgCOmncogalcHJK89jNjQ165YmJKfasold1wYWeQIu//OWvyhNPPMHRkz0wpdx
SAbDEKescvt6k4cOb29hImmukHUOs+yLMhdi4aIkGBt/r6zxZhN3YcBAWjzRGeU51NhvoaMq689Z
qqL4xgRPwPLML1gmRN3CMsZO51+ILYFlzdGyq2Ry51YX9z/zk3/zxX/37f+9vR/vGuv/4T364tfV
v00Hu1tKYtNodGZmaYY42MT7KULDlnNap7V05u7Ejuy7P7OzuSb/XknTMec6RaXokGBm0jnDr4Bn
hCZGTjk9PMW/dPLtF4GhqaoIGCAQYlEJ4zc2dXdZcsTkA7QXgg1wW6g4jrLnOMRw+efKkPPTgw7L
b2mHoC08Jw+R8V+gIg/jQTyu0wbJ+2qyPaO7pjA/vjQ0Ez+M7+3ElpwHTECNFfwF8FX6cJIEkoME
Qc4sUgKJnpZGqqBt0i5XhlEnHeX3k6FDN2Ntp8fhxbiunl4+6cP7nz57deOq/+lv/5YtidMd+GPw
9ejhDm3X52EVxTdX/evCKS8dkY01D0NW1swwbtzuFrG62ZNYZ5Eg8K/Vjb5B89pjUcq1LIicFiIT
Fi+ZueFiIaLdaAI72XNh7gF4QCxYGC8+z47w1qiYLBxZoJJsb2867TjCEPnn8BI0ZIejx48fpJT/
x8U/K6eWTVJqgdCkogL2UzegAebSh3CO+CG2LIhgaa8LMZxN6d0OMlTwxCIbNflbnnUkxdO8GD8/
nnbcFKQLeN9HmVw6V5vt6ehPeg7XcHMOehZKO7byrITQ6izASBALj3V6z1epcudGoffAHf+BNX3f
X7W988N9/+Jl9Y91//H8+nJG8262lRYAo8AJYcHlz3BlBj54QISEU81tuQU5dc4Mc/rGflXxsQVr
Om+xs7Epe22PISIQXNVIXziJZq1E8e4eTyJeWDnKe6rrLQYH2YvHC02AzALC05zwrhk3NzE6RbHD
65CkSJWC0CKkPHDggjz7yRfnCFz9PxQkYLowNIzjQWctWNi/Fgu4a/F7pkIn2xIuCSon3tjBEM1Z
4cfOGyE/r9YhGCNULMqZm51TNEaqNMDqUc2D1+I53doYeE4CKPWNKc2J6e0xwR24fa/1WCnXHaA7
oZig1JU13/V/T6/Wf+Gs/+vYvj4yN/OD73/fv2xfaGtrvZ/0ePJwXiLrtzjNuYV5WbzZkc23Zudo
9WX/6a7J58jvyhSe+Iu+/+08w+lh+8//+gBy65OXsmhmfGGUeeerESRLwL7vsMi7Gne1NiqhRA2l
ji+EwvCi6YNDOBoNGSWRvty1z81PBuKFICOkVdNLAQBvO+M+sLNOIDxw6KCefOy2/8Zu/5cLwVRo
MDBvSMFkWsem9YaMkBz0NaaMicH5NsdDAJZaoMKmutyedbp+bCZ5vNMZcyO2Mtalln5OnT8jM1LS
M1sYki1OPCoOq6DnHsXptAlZ+rVqPLL1tEjmPU2ZzSeIZXxQ/z7XZPvabiPfq4+Oja24zu2NmavJ
/eNfvvyfbz1n3H+Hx8Yfuv8Yt7AUS7QuImHXcgnNuYXxaDlxyhVx11VX0rocvvkyuee2tsuRC2aR
WyDNPP0WE9+hFF8tFF13CnlLkqPBsyF/Xz26yxAK+LlBdLGWEyAyPnaEsLs2TWLC8fFrGJydkdEQ
J+sBTgRgvr6zRUJFTLp9ak/ff+QHZ2j7LXJHUP6ghZpEaZaI8XXwGPCO9X80YRzWVoUFdNY+Yg+J
3bIJPcxIeyMBKmlpjbejx49ymJ6eY56aR1onx6GUKLOW5NrHXkiKQMPBZFI1LVRQuylRxAog3w3L
k1AMlY8SRHpcUcUCWke/2utni+vrGzz13/NS3f+on//rf3jfW/Qcf1732lRdPjE/f67b5qXTQ4dy
Xbn8gK+tbko3NSDRzgOMpXnLpMZleXGAoCBkV5I9o/D5y5Jh6OZczguAAr7m1u+eMapekAXg2eMd
RlHlqMT0tCRAzU9La6xDxNSVCGAdlR52HhUfrd9o0NgBRj3/pCXnqyW9wgXPcBvi7VHTI6TFhtPS
abtGPjSiVEd9RzoGHNqXCBkPZgp4bXzAQgE0oL2EjwbHhuMBBxr/BH+bcHQOWRD0gzhU/w6PmWRx
KOchVAaQVXoiNGwg9robbMFTlK2tIjGYCC51htAid0fWDY9ja3r7s+edPvOutb7rtmy48vm4/Z/0
r+nj5y668zq2y33fGdfHVV189hUX0za88KU888Tl57JGHnYdbI8jz9//bfyYvf+Nfly9++Wuysr0
nuy6XBKHBwstedyATk2NchNtbu2yHQ354YGlBife7W84w51UidKvFGilnozoPCmOjav5um4v10OH
DnFh+4sRx9Z6Sq+DZ+obc8+G7XKib8z0R/sIgVEdJvVXsc1CEkT1sHC607LtjQ9qamygavGwjYn6
q3m4QSi8GVCGHxkYAnWEjRgB8CqCRKIkCfwtDpTcstNUu9SAWjp2G6gzY6s04P0Tf6JXVLh4XHnt
v30t7BLugTwWes4XEeH2322261OFKtwE+dOstNz09MTH6w/fe9+CpfWN9kT+uv/66qNVq/fQg7f4
vSR4dyqNi6hXXvYb54/bmWXnyqa/Id595ynnJnCUV1A0/ds8HZfQnJ+VHfvpn5SMfeo+Mj40x1MX
k8alppRH2vVIgctN6I+HEcXhePDczOyM7uxuS9TI5fPgoPQnC3unpeU4wh3FMTc7I/MKs7O60XD6
6xvwWdERwgLGu3/ve93IBg+TP1jQsfBiNW/C5W9EwDkVwm9rqBoEzF45iQJZpJ+Hv67VmmHsDY4P
i4Qja5pwHBqd4fXWLs3MOHlhkJw7oitgYrKZKGwRjyTnBhvv9INOG+4Z73/5AGwfoaXMJ+k5leKz
DtjBRLyb9MSp1oIBKw7G6a8OmCE5wd965X4QcN897UwpC9b7xuhuv//Lk5NgPPPCxhzv7YfCL7PH
a61+dvOK/ePk/2d7aeqrXbb+ryIorB3k2NeHywptedxNf86UvPsJ6YL05RrAFdUcSG7od+eKnPy7
zl14mf+tnf4GeDvnk5PQEEVrIfgJsYq5Xj0lwgPeDl0NIjA4ZoLFLhw678LpDLwlDxeLfWNa+VGg
Aw9PCkFBLxd/0Wh2GxB+++yP8HHg3bAo5SzNF6J6Bp7Pm8Z4L42G4+JmIbF3RXmws8Ias7NAgChp
ww/8dQn+E9r1BVw4tLfK5OG7QMxK1BYoMRJegmba1ppjk4/YL5rP5MAaE12rDu3rUqjaxkTLAcda
IwL1XX+VkFHiiN+X5Mkx2mwAGfoGbDfp1OshdupLfurm+9fyNN7z2nd//ptuTfWN9MRjpda+qvfp
V1/z63k7rObfgfz1Ne1e6S9rEbr2weIiE+KuvermcOnmcngWe0eqF+BnGMT4yzoWzvHJaZhcPshk
ci6nb1tBwbm6BwtpQuQfX9+jRixja4r3PnD5JTzM7O01pFuSI0EfCdyC+kGopnNfC64iCTozK2so
qFzgAp4999OPyrW9+U+feIPSttKxlvkxCpQeXC2fIPV2+iVwSr4dhaS4IOdGMnF0VAe8wNEXtF0a
/vLpG0gbO5eDBgzwOlSitlXkqFSWaoU5roTeeG/jGeiLAhYQ+2KzIg3EmvuZrx6TnoSqKrA97nnI
HQ6bdhqahtAvj3ebRT/Pw+4HPeUHaqDVGFgdp/nOddu/bt958y3+3HwZfqKDRta9s9NPB77nd+za
3cBedF2ticdTjOnmtE5PTXDSTziBe8pIr5PFHHuFCSpo5yxizc0uytb1B4AUKho0m+kk3GEaeWT7
lQuFJmZ6dInl/d29Tei5cm3He1lBgvAfKNBhngcW/urYpkxMT9DZnnTFOOWOGR93ZRtmkJUsLSwR
VEBJjcBTEzO6/70H5+MMP0dsh30V9ll6OnF9npG4TgHEiV4VxW6gLEgR+pkcDcQGAlDNYbDBCpDj
hcWy6sB/GC9YRar6IAuAlNS9VJUMz1BrE1Dz5nyExOMO+NEWP635PUAkhcb0RCBg1L8LGzNjXZRF
RaK7rji9G7NsgCwpEicwbPY4ZXnu0Oeo2nxrDZLQiEgHPe3wNBOHwGnecl7mvf/3KV77yZ5uN2j9
87AuPf2rfWC8MIx1znuMOt+Be7W74UajOU6SMO3wszdFxF3JO09sg57v66qvp5RCKAQVtb+3IqFv
MzYUFabV3GQq3ujsu5J2huuDm1prLQWdJKgDLCB5k0RkkHiA+bIFL7N4HKOqxI4dlY2tTnn/+eYJ
KMKrTp0/L/OIcvTYMs+Y8E8TO9jpd6bT3fGP5lnzta0/KZx/5nA99u14mVMNekPNRmxxp1kO5w8C

uQEYwjahEAtEeHhUbFkLtVqdNfWCoVEBgTUs9RUB0VSQtCuErqytpQV6wAkU1fT1mzLoUwGbPqjf
WgJDi4chHE5h+g94djCs8n6e+Yd+F2ohiwEnGBjKG2beYe9toaqsduojabR9iRrohFDiuHifwRb6
byG0MTWewV/Z6tQeuuuqqr4+ONv/Gl770lWf2jfX8DHen3c26x3mPq91iWgRwgvAqL7TdCyHg5NS
M9wZpYNdccsklmjvBmBHSuecmnTGuLq/J/MwBWXeGgxaxllvot1x1RJJi14WzA5ZhYIDI/1wYFrw
WapKd7q4ziI5bXNpzSrTXedntrRYNM3eL8/Sp59kWh+MEPREGDk+7vrYhT3/r2/LFLzwhqytnAnA
VRZr70VzR0VOvc3EjnGQ3jDOAtO+M1ueohqbC0DVHFDKSUB2BsaKeiw2jwd7VUR47Dd8LjTP8lSg
IqsHLKnCkWk9gSoHNZVkaCkg1jBUpsiC0hs2AIbH7L8sHAT0eDFQWJsbwaMi0RsIIBueD59Dq107
bOi8ojkLpB+eoIXeJXBdpLxxfonXmpvv817jvT7zsZVd9udlovO1LX/5qa99YzwsjfeXBbDC4qzf
IL3cLdxEghZUaIGqG/lIQCmBcCA1NksQW86jLR3cwb8YZEAxmcnRSlvNYpmYmpb2HFjZxhgpAada
Fum5RjU5IMxrhOAqUW+A5Wc8cneD7biGs7fTkwMFZlxOOMAdEKxsW56EDB1l/3d3ZkHn3czYoZNv
9fODAIhc/yPlYmMdPnJLHvvCoz1MVYTXvqKiv1kvJGqpZCFnQ+ylpQTtdoPgPA4ABYSFDGhUGuDD
jzmWgvwNbiSE0qJR4v6QMf7Ex1D0x33ScOHzAe3MYTcOdO/JHk4/p9z2TCSJrmDeLz+cs2j4nu2e
5gUxK/K81G+zk6fYyz7pKSJ2088WxF54ppeCaoltRkemc20odN6ewm/jn4il33Lf2+vl3r3n51R9
1G+bf/eznHv0Ll0rdB5j+lMeNN1x75FWvfPkjnU73W+1u90bnQRbZp5nUedMtl5pxng5CYyzss6a
XhoWAkYwwtCkyjCakhgUExk+C7pYlabvF0U8zKutj7mmnDxKAkth397pEKMFOQr8n8lv8e2FxVhY
XDsiuM1oQ//daPRoXvObZzXVpd3ZIym9t73BDOYReVhjo8eMcjQHQ6e577uHvSi9XoqjwonqOg6G
wU1vUan4CQEFxcTCUzIgKJelSoXBkbMwboII2NPxCecX6OZEXW4sJVunGFsm5tFfyj0WfQx5LIw8
ibYSUVMcJOSbwdXralO122EDxWnhPSK0SSPIgmtZ/1UNj84VRmrZxof5en6dl5oHPjHPM85jUS5g
NXtPvdRa7vcFPn93Y+tZ11776F37h5/9ptG+s36PHzTfdcNTlpY/s7e19w91UZ6TpVM+5POzu2HE
pJzJQ/aBZMI/qTd5sHRmRBQKAtYAhb8TcUws34X2npyakM9DdGojp2sopGR2fYo6K/BahL4EdD3L
ASFHKiZnDpRoSM7fL5ejRwwx3Uc6BkU3ifc5uEaianZuUDfczyjRTc7POUHfld377d+Xs+oqWZ7g
o+9wccG4Q56ZhpWV+aGwgXczqLUEVhFHSYL3BFOJDW7eakB8C1DLPZSUVGBxz3khC7jpgi50PwYv
Me08VAy98J0/d85HtmODwYm/Xqkih3hHpBNhU2BARtaDrBqAarp95TZSY0kEeQnk2Gehu5I1Wv/I
wICwmG0v88eQ8VzEBcn7HZ/S7vct299q//NGHPv7Vf/Azf+/AvrH+RRrpLTceueH66x7Z3W092W5
3b3Q3bMp2YeYwLo/DLk3CgLuRE2OTRD0NbFFurBfepvqgLiDUNudmp0n3Ix3PhWWTmG3a63CiOIz
3zOl16vbWaiPOo2JIcsTPAQkffOGxUQWU4GFJTOA80xGS4QtPIQSQA08Ow4SXHZ9oMi8GeWFiZlq
WT56SD991t3zr6W+ShKDKginlQ3HslAuNm1yY2CQQsqO0CUPSnDbypZEajSAYc9bXVjjoJ7kNAqE
xNqhRtwlxWHJcDwaHDYyIsq/T5p7KiG4ZC7lx3WCJ7IzLJJRrzLgL77cgy2qGis9VlYmYhli4kDf
14zuMfph7Q0tzzXM1Pxf9wu9IRcw4nR0RgZWOCp8xZ/ZTXhq6bcwcWM2WvH7Tba7XfP6xL37qR3/
kh+r7xvrn/HjTm26au/6GV3+y0+59o91u39jt9aaiXHdNC5fYxpV6QjrZMAnlPW3Eg3gPpZ6gCIs
bHvXbzjgQcqI8Mj417Xb5lOAQyjDTkyq2vbvbkZPHv+u86A7J9zAetLNNTo3TU8BIC7/QsKCAupI
r3FciO9T8dzd3ZbfTdZ72KH+3urou086bwkhOfPc5Of7s8/Lxjz8kaa8LzJMLGYvdBbEyMqKbgIJ
Aykaix0sk5N4KwJSelnlu7jtuEvGEib6cXj7Dv0epxsJcy+GDARSKNsMYYaD9IvMdPJGXgamzZKV
SL/UgE2N/n/hSkg2/wvVC+AtkHohwZ9D3gFPmQaw0vD7z4T3vaX+gqheewcTni9i/zgAwbySo71Z
YU/Z+dkx6a7Sp3t2fK5999ruPvv2HfvDPPST+Kwkwve1tb5zY2t79k263d7XbcRcHXW+YUDHIFeq
3Dg+UNah2HyW+j1PvIPK/pYXFQBKA5IktTFxWeOTjz52gMU+MTMj2zqY06yADoCulybawlVMrMj7
akKy95wwJqhCdIPNJSU+fM3H3d8+326jTDlzYnMmmy1F7/QYpgthAjrhcFoa8tdciS6jd7dHLwaO
8+73vZakIQmMpAJaBTnBrNrQ8k/v8FTVMhuv+PE2Yu1Ef4aYjftobyzn1ctod/g2Ko0md4m9YU/X
GxfesqeEhIkHeH3uWEcj1wSsacR9c4KbWUw20s5+tM6es9apxsfdVLGTXkZL9tOPHcXhFSHQ6FfO
zBABwKxMxbnPO8hvBNEQL9RqPk3/mN2yytDwgp1Pw1IHCGyNaxpAwJBguc3rNiRMnPuh+9RP7nvX
P+Piht7+lefMtN9yztrH5nVarfatbO4sD5pMqIK0ymikRRtwwm9od+bCPYy0KDQfBKlrfWAti1za
YKZIyB4PhPfvss86YFDhCt8vExLQUUDFwoTXyz5dcdinDzrHRcff6jgqSCWlvLmRVhQWAJ5BYQVm
k32tL24XN4AOvrq/JkSOHKCW67gwTxnpgfk62Nlschozyyfv+8P2y7PJiLLhuvxUAMOon+XCUBP2
aL6EYlVDri964BvS0ZdjPlRrC1DMrp/m7pcWDNHCEpamvpRr6a5RBu07MVQuPyuZR8N5kJkV5aFq
3nDl4F9/dY5taj55UPZt25GSlAWV6PzTSUc+Idj0Ca4XKp0a+lFRozYgGaU30FuoWvtQU27EQbk5
CvozXIVrBd/weoFa703n7DTe89tZ9Y/1PfPzYj78jeuPtt/yq8zTPuAX/jl57sKiIbep35cJ51EI
9TlqEBZJ6owPDhYbIEHWUuzDKENoC1mcnh93w8uYXnNWC7hqEhfBWmOpGzV4MJC4AxrTloiNHSR1
MatBUmvZSJhERzJ4zZoA1vb0uO24gH9rrd9z3lnqcQpFLTB3HMaOeCHJF7nKvpvPYD9z/Mao+4Bx
QE2UoL3EYWWEe3IzEIgd8twc9dyIeOS7zVaK6zhhW11f4PvPzi6yvmifS6mgyFMJaKG2tcPCqaEz
geEiQ7ws1qFoyEkomBi5ppCGhE8kYVBnr2AqS2XWv/h3+n8QqW5pHeZkvR3qvjJXBfxO8Ug+rYS7
/WiOmuBQv52aDzcAjyKzBFbUwmxbn5jaK5u5u6w/e8fa3JfvG+h/5ePObb3mHW8BPtfe2fz4bpEd
TD5DQc/X7oYaW5joxXBIJoWfpReIAvOBmwWDHnNdCyMecSfpszjbPgZtnYMRXv/YVbgiQBsUNhrf
D541OjTkvuefCYJe7jo+yYwRGjxwSTBp4SRg2cllsCnudbedRW/R0YCPBMI3WuLgwwxIOAa25Of7
86U99Vu784Ad9HVUFzvKiVHag0dVjT/XLA53QckIsRII8US0gp2YsXNTuc5fPrjA6ABlDNzEJmwD
KWZZj5mkWcsPQOQMHhdJNrrIwocRDr5cGQMtyQzM+3De90F6fGBKnud8QMh0irSykREsxnodsz/F
+en1iU77g+3v0t4iRGsTBY2JXLTyyjftn18FQ7vK7b4wHHJXlflD04LLTp0//2r6x/v88fviHv//
i229/3eOtdveDu629K10o2RwgR4p0MaYeJSRimKJ2GAUBME4u8yRzM0CrK1LusqnN1+FGoQbpQyY

```
bzCQeSTx98gznnS4uzQpUDcemJqXn3h/Nz6PjE5RrabqFLYV2nkTaY0nmERq0EQZv722w0RtoKwn
yhd46GAlAIYTLMOD5OZerOoMHnfATn/gEARdsSGiVQyjXROjrPR0lTjD8KSo9KuuduXbZYGFic9K
mbW9ool00CBdxbOD+QvepXhsNwmcgVjCFYJnDh76e7ECAShI/X1Y1hW38hoW6KEHBWxrDKXhmv3G
UaKw3FKK2qagckzfyJAobLX7mJpXwqNyp4J5mlFMlSpxryMxz9+EzJ7HHXi5VkTbVn4q0nhz7GjJ
rx7ECY5EPiwFS2eaCddTu9H7q9tture8b6//L461vfcM7Nzd2/0O3M0B/IqYSk/0S86ZnbJnirYb
Kez7wzJWIwITldPA4Q2yk5sgQR5aTWiohr+2yJRBSsEzx1FPf0vcstIsFYBa8339wHndhbp7h49j
YVGiqxs2Gh4RH3dxYZVmm3+2ozOeoAj1jE+PScBsGhkWR0bS4JIsHjrBXdW11WU6fWpbnn/sO82q
Ev3CTjWZT64U+pEQt15rKufBE827zrJrnDXw04YkKDfVMnCq3uS5H3WdSdnRUwR5sBCTq1yJP0Uv
O2fRKeZgAxnlvzzwalMJM29qYP3rjNxKD1V5xXAibtUfVPGOpzRR74bTYN71H/ndJomWm2G/KFh7
zc2L1yomyLsLGDShYw+Hcb5ISyjisDRdq9FIMwsYeFeX5unM9un5287f3jfWcxzve8ea522+7+fH
dndbP9fuDqQEL98NIItk5keoCAenFDbdd30ALeolIaXXwZOPU9Y2G4HrcXNUDKqeiRVE1PdGbC9Q
YRrOwtOiNWuuOe+2WHDx0lJTE8YkpGg28N4wRpRfUVkcnJonEKnG94QwHdcseRbknnVfWcYoJPTT
a8PCZ8Ngfvucuabtc18LZWqU8gU2o6ZFU5m/GZPJyKnlRsuZC7dg9Z94XXh86w/OzC9Lw18Vea8Z
S/bdtdghfDWiCoQFt1WMoF7blnXyP3I/iQJRiaoUEfoTemSLfRalbjHsaFnVcCxuEUhtliCGFXNU
UGOE9TeTNFCjsb/lheRRCaNxeOy97jbK7LAyOyVlmJiRZSIXchvlm513/s3PXC65084Y3XBu5hf7
G7l7rX4y5FX3w4NFx1MHQxO0uymSvny/Ck+YqwqPoKuQyU2XXGEqYAgzwtDcarZHI3SUFz7Q22mT
eVX3YDQ/cWb8w7d8hZ4098MBwOJbPf/7zcsP1tyhBwBlIa2tb5haW5PChi9lvunjwIMkKlAt1HjV
NMepwxhnkeFCIgIcEk2lvT/NehooM8xo0cDSSo/H83e9+t2xsrAeApl6zJu2BIrWNEtG1sD7wc/3
iqoJJMZBiX+vFA80DY86jo+WNC4i/U+DGWFow6CqQQ1JJhNazJHhVRDUkRbjnepn+PUC1xJBaboY
JhdHM4JH7RjbdPfYzghB2xlKqQlRGTAbD95usecQ4qtIbY268WtaBV4zoiQeZhsohX48Sj2RXG91
tMyvoTXn9sHb8ZiMeVU7TfHFza+dn3BO//VfCs95yy7XRdde97FdX188+s7F+9k+mZuZuXFg4eM2
g27vM5U6XuQXrvgaLOXRkCw1RsBPnReR3t74ycoqcZRTkqFwYXtNWb2JBr4O80hqZq4Vw+179OXg
SqZcLRiJf2lA+7beffsrlq3Ua5uTsjAtRn5NrXvUqOfaSl8js4gEZmVAtYISmOuhYc9FxaC85I0W
bmQ4kTnQSm3tdo1nj+RjwBdT24Ycflme/83QAzsBKsgWK8wHryYynGppWG75Lj5H44y+BNtAaVYx
tJoSuiUfKzTA05PfoayZDZZYh2qAvDw0qnTiqwB8PbYwG9vHnQtlEWk7xsblHeAvvictatwxFCBY
xiT8O4y5rPlyGwxYC69+Uz4HdZEwoOcdTa5Tl2U3c5HK/2cUWeTXd+vzHfyXC4JtuevWBvb32V3d
bez+/t7N92cGDh5vsBW21pNUBQygjyJJVyi6W3zD0Ah/U1+5YiPehDkszqQ/LaqrggNw0y0p9n+p
NOXeBm7Eqpc2rAdquberw7l499vnH+drJ8RnOtmkNOnL55dfI0sGjMuXyVoS6oBaynzTvs5QD8Ec
R1IR9lZwG3mwSDTbhMmNM4XWHXUj9hS98gV4N64MC3D40rPm5MkbXO5dVdO7GY6WSEA66xQdDxWe
hkaAa7prhV9lFcSKha8UMylhL9LKIQvLII7P1oU2R3OMiDqFvIP/HWiLBRlijf/X1VPHIciVNsfs
fwtUoGhrUbPKpoSyEex/Fobab+/CXDQMobfFLN29YZjXk12vk3i8ajsKAUTBw9wCku3YHX/2qV4y
/qI3l5puvu7LT6T2+1969xnnQ5uzsIj0Gam0wQiXNZ6oMUGnBAnUP+RkU73lDRAn2Vkc0pbxUyht
MRbxaIyzoquxIqK/5f5eopC9nRIoa8neF5TFCbw05UOSaaAyHxRw7dgnRYRgchMzwPgsHDjKcjT2
pAItW2Tz6ngh98TqEn1icVkbC+Z4+fVLe9a53yZkzJyphZhyYOQh/oUNs4maWlxpvdyhCqBiGlVu
wAYBiCama0Enj/7bba5cbWVHmqGQB+VxVUd6Svmikf3J+ayXpwfJYvq4WaYSAnDHKK0OY3QaTFEN
GHELVIg5lFNtw9PniBfcxqpDzbRNHiYvhea7IPDxuNuh7MoW+T02UTFH12ASgojSUhhTwGgTPip/
deUHb6R+/aI31DW947Vi/072v0+0eRc0RHmjMt2CpsWiIZ2wZgkV+50a+2ScXNOeF7g7SACaR9ZJ
pfyJRRW98I80xT08rL00VSLBFXQp0ZaGQH3k0WXd7RZdjMXJFIffffz/pa+AIH3E5Kjtp3O4+Mzv
PRTo6PsJIYengIRotpFnarT6HVqFkg9fjmAB4lV4v59889tgX/fuXIWWWK7OKM1Ch6JCVSK/VLy0
crXoJM1Ao/rOVzhk/dImnZ+aoRhiMJo+GjJ1lFQ/4WL4cV/SALQJgrhlFQWXQrqkZHTpj8D52jxU
RLoE80AsZLkc1zVd9dGAePOAGIkP14RJTKIKR4r4U3oCDMHisPbRWekKWjLQJGzCbHzIdzsWWOd8
aqABUJWzWTLiCdfB4mm6d/vSL1ljb7e5D7V73MjRoY5edcQs41PAi6/IX3mA8n/rcSNkkuacLeug
/KvOnzLeokYVipOxCmT3GiT03LzGjDQuIjJo4UOOGmDO596xxEVDiD33oQ0Rz5+YOydLRywgIoYE
ceSRKMKr4V2P4C4/abKixkG0EL1tvBp1dG/iEQ9nc3JI73n8nmU4kaGSDwKRJ6koKYD5X03CVYWj
qSfKhCbsYyvfMmOG9QSVEe9vU1AyNIYS8tSgYAY0+10jHNrIkyH9KyDHJAa6UbWisaZkz270N6G8
R+RqtD1mjmr8Hw/fBVAyN5VQdr6GRjhdEGwyG2uMKgoxJyKktH7W6tIKPXvtYUl9z1gHRVuYjeUR
sYynz/SLKfU22ETp53GP2huuvS150xnrj6659h7s4ryZJvtMhKUASC5Mk3ExV2tPWLdABsfPZQlQ
pEUUMG4mqAVRBAWru5JrzIKcyRktZa0yGSjPVckDZeeHzSo8ActQv8houSFON71Pl71d+5VcIDB0
6ctgZ6TiNEIR1GLGGjVFQo7ednaBSXVvjSFaIS45ut9OXz372M3Li5HfDqAnzZNqipoaKiW2sH8Z
KzTOjqVL4qnNk2FPqFhpUCXudLumSFDIb5F4su0wJlEeNDWA4byy5xTW9vt54w2CpuM5UBeg0w9M
CqQsa+4tgsDZZXSJvDDh23tO8EspK8PTmqav3zDZYgl+e5C+e76sEmEHoV418uS3GZsExARZF9Ms
SkEjo302szp5Hw17buMZ87SCwqdzvpnr99OCLzljzNPtXLi9tIgQhEWB8mjVT8XqySmzIeKFw/at
c0tyK+8ZOqXvZjiINO2Z17AJ6STmuIfKUM18PjeMyjKveeJu3wsVRKPJHipkoAj1cLtB6LX7/wAM
Pyjv/zb/WDp3Egxc1bVFDvgRPZvleaL524TwaAOh5vSqDhcArJ1fkw3ffHQgag8q8VOS9rLHWtau
F51SpKcNY7N9WI7QFz03OWRaID1BQbLqNpVoSqfJ9EdZmeZ/lG6gJVj2X1jS98cb10FJoZZ66Jz7
wc3H8fU1NkjBIWUkRMPKSb12plyYxja2MaDxqDEaae0+8X5nW6HLPGUL7lCUTz1zyYFVcK3tmC2W
iRawL1/xaioY6fyzKkCIL3t4UQ6IiHyZQ6PFPua9ffFEZ6+tvuv4fdfv9S7hgKS2pXsUYJODaYjf
```

```
UQnstlCqsJxIXvyorQvqgD3VpxOg/jSTA/VSPbzbCuIhq3mPT0bRNzgriFSPOxIfDJew/tOnkeuN
TP0jpqae+Eco/qKsOWF8dpabt1PQsm85R5mm5HB3K+SgtjbtwufD83ciHZnjfO+/6I8xroReCprB
1mIAlZKUR25iqi0vZOdrmZmE/v9cVmMPfPv/8d2V8dIw5snoVj5pXWEkcUuXC9ixVgAXpBzYaPEc
v6gM+C9v1uyK4WablJFVuKCqpRGrkpRBCW14tkdVNff2WgnWp5qqRYheMeCIfCntJiSKTAGpxaLp
RSOt+A/KblKH3Csp5+iCkSr0njnxEZp1YVeeQ5fnwPY/85sD7PwhKIu71t7xojPX733TrvFtc/6u
7WE0k+kAiVRg6DjxTekWfs+V+EreGm6myWxBKJX6SmRQVKF/Y4cH6q/cqlPIo0G+pYxn0huWeYiZ
DobCGmekQgBG8rUmc2Feo8ensUmteP3bsIuZ/MDiUn6zli6MJ3euBrsIg0dsKsj5KNcbRRR4b+c/
79reelY9+9F7NsdxrEWdY+Duk+OANq5onktqH0k8+GOqEwcaIPH55ZYULH+EvJ8D53NvCRer4oq0
wIM5aKlIpz2Ko/GObXOHzOpSRsLnWG7HKp6CsU6vUe0OombG8GVIO7/0YHRX6d2QKZRKohmWdOA/
AISmBNSunqcGixs65rsxBtaHeelUpI+spigYiJb6mHPJxH2X085KqKP6e21fk3yuKCr+urCFCxl8
Uxvobv/GrkfMwD7uTOkoCuhe5Aq+VVLBYvSl0dvqDsteRuYbvWQy9pdT7iaGP4zsmIhq1TWIwPR8
suphzRhtB+b3KeDGjNQPVxvJiyIBNnQ830Yw2Ng9duXF462PHjnlpzMQrMyjfGHNjIO9CKdNYwzO
T0IyU5+Y/q+AG9r+/8//QNjogtrlyVGEI5OY6T4m/pVHU60PsHTMipBHwyPCKJUnBE/R3t2UOdML
GSCA/IBQMORo8nNSHcnfdEOpkPIWmcYmC1zavFXmONFQH+beehQQJUd5LjyBz0xWfHxsQZVFFhex
CIMcMhZFY4RvE+yWDyjjAhbYfgr0W0hWf9tjdNS9aVNKYoig7b0LJDhGX/6sgHicGaKU+SlOBPAJ
UReATN256/Y21C95YH3zoE7/nrsM1DBkHKjoGAxsZGfd9g9EQ8lptTA4XPPck7agWclJQ7qrME44
1dB6WpHEsHE9AYF5Wq/khwXHIT6t1yCrSaARyM4BqCHau5zXk9IrLr+Tnow1LF3NCIAdDh3e2tn0
nTUKEWAkGiYJK/j0RCXz0/o/JU08+SUK9Cpk5L9oYpfeG19McOKUnDuSHClNJNxcNa6PY56CivGS
QH8anZ2TC13Uth7V81cLg3HvyLLCAsNBzJV8YH9ir7AfAjvIo+VCtWu9fRMV8eE/SFIuUobFFS+U
GUQKL1ssaVe69dT5Zy1vkaY0hVKXaWhRKfrqBWC00DrV3y3PDRmwsKkiwSjHkye3cxc+rDU0RkYT
KAJvUo/CemMI+fUEb6y/90j+7ttPq/BSJDiivVICBkHtl6iGzgYSZMZYqcFfNNXfxfoRh2YjPnzR
3UGTUdnTzpFaqsTCsRPNisZkpVQ2hsFkUmrswzPVyIKETx4y0KPziy5mLQuuXuVSkqvHwfpsb69o
wMDlBAj9MCXmsDhIujxOPs+ub8r73vS9oQIWm7rTngaUkPFfVUEoqSPbAq+xbfZTlF/c5Z5ZXCWh
NT0wOMZiMimkbox3XwCvlJ7w3fSViiAJNMDzb2HQwc0ll5Cac5aG0UxQlh5geKNOYJGEfmxZOJDe
dJFEmU+jaqTCdJC+nxPlrEyf1YXSW965cXCWdUfPpvCgq2sV63ULdGOdjqHOahm6ec0klADRDGiB
ZAJ0Qubjfo2B+5QVrrL/zO++sP/rZz93pLmkToR3VAUXVBLW3skY6mAqY1fyks4i6u7bLhZqh6O6
d5VqY7gHc8WEySzxuUVsOFEsytPPjMajUHqvyosqKiYdYS4UPxXKvOGsASDUM5uL02rRoVofOMNh
XCFPxWpAaAPQATJqZmaIXbeJ3knKEBTedrFwIDz30kKyuLTMEhndE2afwYyRU+GwYtQ2GW+Hk1i1
PjeoKNLl/L6+tslY7PzcnI/VGMB6r7Ya5qHkZ1TDUKzQ8jTwImGeVOTUeoSUqy3psXZlmHhwsqY4
KMgU0nfdQSQqpR9pt81EgL61QISP/uRIYYwp++dY4lmzUGBNPKdTj1o1WCiO65JVwOB8aERKIMD5
XtmYDqahGhOhFXb9O4Av3oeQow7O6FO5HL1hjvetD93yqM8guM31X242xoJoNBWEQFquAmWeaJLq
QIyn7HG3XrmrtEOkrTAe3V3rrKA6gh8mYUFmvWhoIBpcHgKDqYRlKGj0vyl+ABFeRZYTWR44cIQM
L52mKgghlwVqCx0foGzwAhkXlfW1v8+cIFtedd97JcBWzWDAbZuCBKzNIRgyVsLy6kZVIsErUqL1
FbCKHd1+cX/KMJ11oVvIJ5Qq3cvtk/cShRY1ADze+F3YnmYbyIO1op1Def0FqwGOgpRXByxeecyw
V9hh7ZXEaRRyE3DSXzkO9GE3y2IDsnpsMC4P8SOvf5BYD1GskgVvMhnHPPLPPwgZEr+y55Il50KI
SbVjpxm8MWsoqgTxTg9RyWy0ATe51r78gjfUtb77tf+r1uq/GBSaa664Nxk1kmeYzaGwOIU8cBQi
e5RfPcNHFqPWzMvQsfx5uYbO+TYXbKX/plQWMg1qlwRWmJlDEQ0aoi0BDTuoAZTKk+yPew2uLlG5
CBw4cCKWQyEuEaCP4GMsmIEmwzJP1BDAKhhdrzU7lTD70R3fLqdMn1PuD8FB4FQPklNY7WouCqkH
pfTTKMLDLas1ad07JLYZiPwyKRhqXWkNWlzXPxRpmZGFnQqAo9/eGsqGeZUSerrunIG5E0tD3YR3
V18fFpHOGO38ism+9YJuXdVXFlYKGSaYSasRSUkxLQTVciyz8vXXxmCQqjdorO1jdOqIkhIGBaqS
5b4qPI/WeJefYk11kuDTkFpDX6ypCV08o8URF1ata6tK44Iz1p37qx6LBIP1HWVE0bRciLzMvieZ
YyNpHmFXCi6aSrkWlQRiemkZt4OemSlWrx0HX1cgDVue0NjDbgbVnsQhk/qEOm6JEf0NXj+fCMgy
Ky06bOCy8YdHvK664QtUm4lI5r1Ef8/mkIoZ6/JEz6jlo5ild0j2PuTfvec97ArvINgMFlxrB8K0
TxhhKxhaifKgHhxieuXQAnmh1bY3HsTA/z4VIj5+UQtyJ12giz5dyKEV4jh4/0mNO6lqnNn4uzpH
KjLiHY6O+BpqXm5lv1E+ScgnqZ2aezVXT9oswoqMIm4DVRoGo15N62fKHRoxkxH9GVnKTJQpAmgQ
/KAH1jXNT4ddeVza8czi0L9vVy/w9qkYFvg4MJJ6ls6gIUYt1HVUbCyospwvPWF34hTvWKFvRlHk
ieUlnqzdHgiK79kpmFUpX2esYi3I1la+ZE+UjCcKHVrkvh5RIcuL1lJoBwo/ERj7koUYq54R35nH
sPcyrcBGJF0s7R+TL1CleesX3adhVCe+M4WNoY2gA6Lcwm9AtmCYN9I73vk82NtdL4TAPcFCUu+8
FyXxYhw3PrpdtTudygbFJ7e5tsUwDBBrXwRQl4sqwKssdk0I7U7I88i1wWVj4CBdr3iAiXw4zDSL
11DGbEYoAs0aBgGKobxlSii8DMXPVrCTKQ9tcVRKVTe/1RMGtP+U9QtmJ9M9apXNHm/HpDbOS3KI
6TgWJE/TEElXKN1hDkS8pSdioNHQuSGelBwdjTqIh9lIUJedOaL/wjPWeex5I3cls0j6gs9vPg7x
In4X6JAhnVbmrWmyOwoIsYfg8hHl4P+UM97lr1uOSN2qfQZQ1aMTmoevCjLSkGeYq11Gpv1YsOHg
UZltReassl8ZNBbf58OHD7PgZJbjk56p4aiRzSN5c5ylHpiRpuq94lIZ96uSy3PORjwS1f+Sp11F
jXq9szVIVPgvrrDOFzdOZRiIAXGDQEObWhoIJ32lTG0K+uSkwdHVRCkkGvpTj8zqldMaURkmzTsk
gK/RaQ7pmlMQM/7c+F9aFG/P+auiah5y/mv8bgwm9pEE+VPwmJPEQH5fGFXvRtGrHi/hNOq928lT
I/OoTQ0koDuGqD9Zxzf2GkviNw+b7BAVEkw7ytdsoKfuAq6lTXhl2dWECTLH895hea7o5ZMcgTGX
Y1gh1q7wyvsBApCEASJKgxq7lFvGLQskReSi5RAokidLT6l4ErRpqGWhgtcChdrkirpRvsrBAjQ9
```

```
chMkoUmnTSimMBgWIdNALukNUwa/Vw0ITSUMjdeyJ53jPP/mTP6aiPkkenlppA5vs56qRGYuJixd
hcE3zbqQE9vyJE89TywnGqp9fV1JHhQfNvNHnllGubJ6opmURiUuOsPGz+bkut8YsH2hPwUjU+Hw
7otY6+KUpSi10Sdk8HEXWrXSSB9aS7YuGHheWesRZUD0k/zgtynwW9y9TqqdNcKfB+u82PgMItpV
+SnDMR08VNpNJsAa0Px/uyirJJ2lolysjrLzSTldcmMY60qx/0p3QKV6EKOYNNiOpaslaUb2ksw0
3f5sximeZSGihS3whX0LoSxqi9xwI/0JIW/Hg54a/WkwvhoS5Avul0oXCdijb8UVCbqwTySdLAEY
0NBybbFaG+RbsymGZg+WITNZWz8of//Ef89h6qbZ4NYKhZmGMhAE4VkeFt6PH8BPSrPaIzWltfZ1
GMT87x0iCqoqV0LacSVppso4bOkIEJTRRwC33rDIYB8PsQe4jHaY4Mj0545H5iJxfZSRpaYMMqHN
Rdyx+9hgnobtKNXyzkCbpta4OdE6DBGnh6YvmWVPfh2o19iwblBtsEQ13UHnlCk21Co/++qZzL9O
SSVl1CO9hTQEENZOhur91ZNmGbU0Y56ZXF4yxuovj1krRwCapdTDNKcxYrQhOz5HEYWZnSXWLwrw
WKUzXp+yoSSrF+9C6lXpwSZKhXObcx7DI9HA4bvWzaluYlWmqryPN0C2fSy65uOK9lTFEFXyX66D
uWhUcs3odEeAPfUhaezuhc8N4vlbwD721RRzYUyq1GfmZNPGQGl+7vSdbZ9epMUwE2usAczZMpZH
B5HDsuvWx8CO9B8jziIpmqvJobWLIzcFtRjmq3jS5l4IsJ4rWJV6TF+FwnHPYVhEMNfPEf0+GkEo
DeRYP9dJat0/kkVrxZAjlCXuU26cnZRmlpzxn66hBqBuF9p3Qe2z1JK4eb9ws22GSAZxHXoJk3Ih
i5ZcXRRby15JgY8oRGecMVYQM6heksbY7nX/qjG6R8ipxLZRvqqGpkQ74eyygfLi7JXA/vffLPVh
Q9ciB5G7lgAjex0LbWlCeN25vNR8KbKVKV825xPhqTlKVsjSu6NEjF6vRJFaG0PAMC0UbuSs0Rh+
KY5bN3Xffrc0KHi2HMfcHgyEUmMLlJOWXfarYkMA7rioX4nhPnDopU86j6siORkXhcKAgEl6XKVB
lXTOIRBRA095fGrdnYVnzuB3jgAO09ihVqu0zCgRFRYWfTJGzUnY0YpqShOaKnLXiNJRqzDufq9Z
hPb85S1juehZqkIXPeckFz/Ny0lvkJ8j5GTg1Ke9ZNceMLOY2HjX5595AcX08/lHorVRh8JoPs6M
S5EIva856f6OysfP4BxeUsd76htdFr73+VW9O0/yfg9VhpQDL14aUBTHTxQTMPApsCC5rrrm+pmw
WzgO7xDRyonojeFg8gF9CrCvIVZrOT1yEsCurNDHnXgLTmo8t/z13toqFPWWXjjY5X3rppQxT4Um
NOmfdHXouacVrxPS2mKkKr9rzOkAGAMXeaMPQJ3iNQucSkuEVmrz9qMa+ns+pUyfIeJqemtI6ZKg
pKhoK41ZxORlSyY8yLY9hM9CmhyS0ChqtECLqKNWAkYV/g6GlZaKaYtSJcrAtHzwXuUUeWvaL9ku
1iKwMk403bDVaI8iYR4MHptSK9/KRIk5MCSKWZXy7G0k1SclAKkqViaE8s1LOK3zrm1QI+8ZqKnx
kwTC3kCoP+Bydrigg1BeMsd52+80Pdnrd424xfMTtuFNxVkLq1ZDNAABc8EasOVfkCdOaG6VeDaE
fPJ+WHPLQ1qWDmNzCGajHxm2DRx34sYLaxZ8NXVDzcvEQ8psbkZVeuQhesxhSzjsXO8j96IkjR46
xWwazbCwvDz23GBs50NGKnOTmfr+ysir33XefL/so9U9FvdIALlVJDxRXkyKEpZyj5NXicX4AfGB
MaHuDsHi9VkqQYHo3o5I8HWri1s0RuamCIoUnPmDhc/CxkQhITOgz/AW4ZOJueeTXuyc/WA3YZHe
Y3rAGnSv3QLIhgjynpQcU3jOmirgMVc0QRBVBCp9Xkl1lav2Jz0OjqDICwwNviZbeikiCXIttGCX
67DGLRNeAAYmq2ZyECoSWFP08nExe0MgxXHGILwxj/Tt/52/e1ut1b8aQKJfPNEFuyCQaah6veb2
g1OehOp3N9IVUAkTDEJ3uZaWbEDYXUZijwrICB+dqQzplRHyiz9EM1VKFb6EaCoPzqAQTfA4EKZi
hzpsK3fHc8Fk8CLR4YMkr36sSInR+ScEDjbKmiCS2ZeURFxx+vNfa0WjAi4+hnqg7c16CSn5MBQg
aPP+ssmkxTNVa6/L6iswvLrCHVo2iLNPoMQ9Cw7h+lq95Wy7IPlYJPaZGmMfnZYOMQ6KRr+I9gTI
zEgjdK56vG8lQp40ahtZtLSeVMhD10ioSyBNMgXzUE1fb/nAsRpgpdF6QAYG2KVWjFst3q22Qeh8
HYdp5lTSh7W9JUIWwI4y9dE9UkWKVEBpHlVJgNGRuf1bv+j031pPPH3+nM7omZVDIti98HStW4Kf
wLVCeS5kHnY1KjczfiDgyT1cvb4hR2ST2YxRqAZQCM4nQvh+HAI+bxFJpq8o8saYMo+MKcKcqFBL
qmNWdN1ATfd3WygR4r+mZSZmcmB2qCdqEM5teluVxoKohbP3whz9MT1jlmaKjRlFfDYmZn6IMFUg
dic+Hh9k6x08+L+PNEZkYm1QedFxVZhCS67MUx1LzQ421lxRG0CBRYhAWvY2YKAXJAOANXNTQIYh
DckWtNkS/zKVsdtCNtVTMN5pk2ZCRhWtYlmzK7qGiMmNIPBcax2R0P0QtWZoO4RlUshCdXsBIBRt
wosaoEZgpE1qtW0Ljg5H7yzEZ8XCdNi5C+mYpkK7POJhXdVK6P5f+eW+sP/bXfuTV/bS4WJN9L+U
ILxkNC5KBqokQKEeCnvnpbHlVH6nwzyuQwvAnV+ZMt5MGNUNcbFvwBrbQU8TlOMHqTS9R3/J4zhX
6thJDNZR5AVFC95bQJTI/P09Di3w4GfshSGzxK8rGAwtB7733XllZPTPkPU1ehnXUenlcia97hs6
YuBIiu+sDkGrQ68oilP9dHhmUHga5RzTz0JNKBQV4PG349BuZoe/DfGk9rp776uqkOnQBmb5xbsq
H1nnjq4wVvWXqG5FZlKjYWhEPgXwmsEYjTyQwnQx4DAaQWDeWhufWf0yyCfAMr2SZFrkfQiahUcB
KNlbaGwpRI3MMaaVuXg7yChuMiasFj1yqY1alX83DmubVeW+sm9sb74bYMYdFeQkV9q4GsbPIy2W
oLAu0lqyzRQ2yFC8D68U4u3gei2WPodieGkWWByU7k28xbzBa94yYotR8DdIleSlNaqG2hseFJ9X
HoaHAvMwLpVwiDmA2w0ZrXOTLAzh+hMA1L4saxiqKLqzcpQV33PEBFfz2dT9jK5lsiqLa+ZBWsc2
+YTO65wa39rZwzeXgoWMEd8LmY3ROUSG3JDSL93UkqT92at9G1Xx8uOsIqS76aDnw2eWqeLCWHHI
+CXlxVJ2v47tf6mxqL0W6DUxkKuS9b+JLIBkJExKiK2sSJ70RvOM8LllFUTmIywgrifiIw9fBI8+
ttrA2OIG4lENl6uOVLUzW1hrbKVIQsAzrvqqFCQMlUaIoc12/Nur15vntWX/wB95ykQvvLlYtooi
c3apIlcHy2JVIXuDEsSLQ4DKPgpYizJorZGUnA3NTvC9yRF3o+jfKxCkV6XlzI1+SyId5vCqQVfO
0vXroYdRcuhxJwXDd6GVFBfAYEv/S3y8dOERwaXxkVAXeqC4Y+5YpkU67RwOE6NlnHv20nDp9vPT
QFZlLTlaL6y8QG4emsNUeTSYTrz9z5gy9+vjYmObFRUkeSTNVeUjzPGBnuW8z44AKT35AWGqgjk1
rs1wVGwKGNPc8q4oocHNMa6t+7mrqgbQ4lEl0AFRJzPchaq0eiP7ihdsthAw19cxrK6WVWrcn68N
zxlE2VKpCKmX6XCqUV4bbRZi5mlTa7WKqGRYhzC+G0OHcR0CWs2pKNVybjkPZUUNrw1/w3YDIPy0
SO6+MtdXZ+4C7gFMFu0gS38EvoTex8PzKIEEZ18PNrIahyueNwtgDvB5hGIniPuwkl9S3OGmeop0
cGOrUt0K5DY6KhpE7vRMe7bWZm3nJSWVolkVhfmhRQRNjr8VkWkBmNJifqrqftaCMWKKDpdIA3v+
OO+6Ufi/17Jw+vWmjloR+UtvUkEPmYsoZaTmGwjdxr6ysMCSdnZ7RRSL6utgrG2Jh6wjGmE0O4Vw
```

```
k8zS/ghum5fs+rq+EuFnQyLISCtQaKRXraYJ935xu7KPY59a8dnEUhizHceHrropZWPg4LFSnFEN
GY1kp0q6YRkTuN9MEv2EO0UclC+MvrNUxDXlkMbSx5p6GGPmm9aLyczAZa+wvPFOMm4Y2SgRpl7g
2xGQq5W1prE+ft8b6A29906ILaS/Pcwn5YDlGz7yBhG4VBY+igNaGkx6SGSmlO4CSUuTaQ/9W2pC
o1IRt2EyXTOF/CGdz3oqUw3jDrU2jIRqjeqzSWCQepiUWck7+6huzLWw/dOhIgPfDgvDdIKWxx/L
ww5+Up59+mh7JaqjNWp0SKDbFG+ca21jESidKWZNOSfXr9NrMU+ve63I+eIVIDtIBjtwlELwpbOJ
GiBrCRyW513wIrOFpSVIAMo/36XXb7C0m1jA6wWiGERI6YsIkrKhUELRumLxgqDtgmB8HNcKsGO5
1tVTASjiZGa7EYYq9z0J1Xfl+W1YDBmWPadDFsnp6pf5ZnVRXnEtbJT6ShQb0PIiCZ0NlPuNV55X
pd/Z7G5/pP6/nzuf+89ZY293OPe4CLCK0SSsnoaFHLchHVj0cRcsa9SEV/cJ3Qxh5YeC1eMXrxdq
OGhQOooIgCHdvXxPDDW42at4DRuxlLHzdjVzXvGz1stDT6mbokRWP9GmEUNIareRTzX910WXUXTI
gA58JtDeowBfaa4vj/9jHHmCEUCLJMgToKDHdt4gFuZasjEQQsbj3W1lblkMHDpc8a+RhnEnrCSR
pGGkgdamIpnmDyr1GLkA+NnLXyvEWrKn6gWDACUqxdJGx5kiZv3thspiN7FJBRX0Duve+uilFYbR
FdR2Exo28klfGJUX0XF2sId1nOtwsTHdPPeMoOofzHVQrC4/GV4cyGzmnQrzgRlCPy+FaHtcoFUr
ywAWu8oQtSuing16c5I+cl8b61re8cdQtzstNsiXzHTXGkTX0tfS6WUA+xZA9IL15FNrjkNoQJMn
TgKbyYokn++dRaELH+2g3yMB7WN+i5ZZp5OU7cpt4nZe9shb+BQVSzj9JAthk4bjdVB3JEA/t0EC
qQZwHEYEC2ZEyk8bGR0KopRrGEP9+Sp588knN8zyohMVhvakG2qBOrJzfeqjtGdUPRnDi9ClZmJt
nCIy5OeblygWvnS+UZpHIS7RgTpC1sSG39bItUgQPFgTR8V5+grzem35lyHSTPajm+bT2raJnkSf
W49+sDdeioaFYqiLh9Y0y8Z08lWPOFUUvfDOD5MM9xkR+Sa6Iw7BmQ4S5xnx4W+UXE5vIPEXURwy
RDbvyeTVxERmOonBfiXsUeaUUV6Y31s9bFNk5HVzcmHujoyPL56WxOq/6O/CquCZEO/3gJBgd29Q
qDJWQePu8ohaVQ6ViLwFp4RkMlTlk3+9ogzT0blp/J8kRqCt6aL3hpTmVKSXaqxipNzrXixnVjeU
LtmIlwyPqw/j6imZTJTy2G4fpcJBMMQkRYyCB7YNj29vtSFyvyT3orClUd4qhrXV9FCVlMOzycTy
UTkCGFKe4uXWW4f7s9FzZKZQbY6ePWCB0vTBvy9UrWIOEje5IvUETYYZPSTM/MjJWD42Ixv0O09j
NM1UHW4UZqFIOWjYhOimGm82ViVaEYoZhFWH6XNbXUlJkNVmPZNfioZa1ktmkLCJGD7nxxHWGjaU
61alzmVeeNITZojpjSXEjztQ7s4TGNKkfrot4R6HXc+BDdXlBWE3ihVIRezMzM53zzljf/JbbE3c
it5ligXJzc6+o74Ek8fQuu0B5FIrhsdcpoq0Uyh224U8GSA2KzBOoKwJlWXUOSeFbowoaP4wdRgv
lQA11lbhgRfzYS2HaRTZ9JVOE4G6ap5VWOZ8LSRZKIkVetvlNTky7P+5zqjk3CJ8fWa8qjLa125L
PfeYzXPwWIbB+nGndlIvEj65g+FtRL9TQqkfPsb29SdUHY8i0+wMaQZaX4Jwee0YVfIJsCa6L1lU
LAnK4vql6X48BKDFA2xXt+M8NgY3IruLlHlyJcH5dH/ai77ah7XWJbgRZmHqeqUHy3hXDUwRswrl
nHenG7fEMU6uIPHnFRzgs1PjGcs7iyZRoUc0/y1A1H6qVF8UgzNo18j527CLPA5nGvG5uFYnQJyt
DTQE2sMwIE8BUGrV6/+JjF+fnnbH2er2fcVd3kb2bmXFOy/AqowxL37M7ywlqRh20uqF11NiOm3l
pT3qGLK8gyL5zpbLDM0x0fwpCAOVhogbDwyG2Ua6yKxypIEUg9ds80WC4WRbGBA4JZtsksahsCrC
FNDc/4469wfKKDdm14b02W5bN5a2WotZWziq0Bc7YPlU9pbxQcCZo5rr3OX78OZlfWGLpxMS180E
WQsbcz+QxUArnwPM3Aj12/hzhrWIEnn4f5rUasyr1g8Kqhlp6xJJ+aTlo4vWcGo1muN6ss2daxon
jsrcYuSA2EhNUCyUYAl9+FEih9EJcYzv+OiKmSreMXtdB4F/bZm0loKqCptVls9wAxtpQM3nqlSI
IElW6frStMA8oc9AtRqQWFWJ6X9aFhZo1rsPo6PjgN3/r3xTnnbFy0nOWN4m49XtKBfNAkIEL2i0
y0HDYX5Bzux/iKHsBxG6oo4UxNOJwg6tqCwXlOEJeldTC0GUJFzsOeSqL6IWG4mmk4RXD4KIy8Tr
OQ7uTNboTzSxeWD+DZ028qr7mTsNqE8hh77vvfpIL8HHoBjLvpeSHwdDMGG4eJkfiz2ltbY2h9uT
EmC5if05QyI8qotbYEDPfQ6qRjlu8IJ+IV4FIyiYK26RM55eq+dRjKkLZpniBbKsNps4q0+zqDJ+
tOd3AuuFhWWV5ru+nwhmbiR6RLXmZNxzdWIzpxBDeTwc0YoqOjswrOaOE8RV5lcASq8czKRdqdtU
NF8mH0pnqNMFAcIgbXvU/1k3Eh3WhGaAiSIBgB2uv2WzunXddN294wy0XuxO7eFCU3Q+hiwU7mak
RIJTNlR9sOkRD6uaRqvsZ8dxCDLSFqbJAWc+0KXG4fn2O6it3NrvIlPukmHQ21F0TiBG5D3URApr
mji+dmAZtaDh/gTxpddK2LrrJ6alQgCeBo8gDMR8G8vAnHpKNjbNhhku/KPtWjYxv+auVpGpeigZ
G0+m0ZLe1Iwc8S4rhZVoE4I1jGE3flzKdyLtU5zeJMt8lFJXX2o/jKIc8VaIKdx1Y+8XokYpYXeS
NOw6tYzElW2LfSsdcPR8SDNPxIfE5Df15FDSJFRgqQipknopcXKDBHrXXEXE654jkGk/IsL9RYE2
PIegIJ4mXbK2pPrBvVRSCgGUtVefwZOXcVh+1WbpmzCTq3CVxuIYm5ar6wzV6VVwLrD23TT39Z7W
p2l+UsboLdoc74SmyfAqrWUog1GOREfTh0il3LyqFBONSmJ6TusnwEV8yGUgqVZXDIhDxjaAQNJF
svIRJnXoAhwLZvmRiIt+6cCvUMFEPFSUVzmgcleFnJEMlhCqBw4wSYI8BVr12jxxhzlH1RfT77ru
XdD1TniDripzcGkPCeq3ux1s0Qh3aPCcMcXl5WQ4fPBB2cxAqPGuSebFqO1lpoySXU/AMn1Gz9/N
13xQ5fZ30PhiUjc4wI8PiLztnJKgm6KJNFIOoJZUOqDjk6ipJg95X9dAwNpuuZx051FbCdcf5pv0
APhbeIHEatVipgiiHN6yklWpkhPVTnTNbCnhr2E9GkTe6qJ6qWogfhE0esM9XiWoX/TJq8LX4KNB
JAbbp31MBw9f3LT+1EDhx6xdGCmfjvOqO23R/67wy1jfd/kYUto7lRVmvtEVGkSy/ywLMYC+ldcS
AYeMnYZf8Tm2MzpNKXgBjySSIZzMD9kN3oRtrU9IisKOyInCEsZMy3GQrXmUwcFHpqrGBQibsbGp
2WVoJhyqIoge07EqGOiOpgRE9Kwvm7rPbnV2Gqi1ntJNTY/LVr3yNxmbnaxMDcC2I/ia6U5O5U4t
sYKmWjtxxLZ9ZlsnJSZcHTVY2Lj+9nOEk9ec94prQ8AdQlIj8uEV6WIirqdA596JEhmbBxBVKJ7G
CtB9aFYc7F7TlkKF7nvnUwpQSlD1Fih4AJFEtX4Jn/Z6PFnwph17el1R47TFxzkUcvYE2jsdld1J
NtddkkOpGk3iwkb288Yg73yggsyDBpJ4hxU3JHVcjGVUwSnyPrp8+yCOkDlPkQSJ3Xk1VD03q5bi
SghUMCWr71QjLhngzvUZrZq0JY+65VPDr55Wx9tPe2wdFvlhkurvYBVPjaLoF09PdDeERRKt9iEt
QJSstgABHLAGYUINPqfOgwFMF3PFhUy0pgpejHGmiuyJC7Toc3AAk7HiYEG6epyjV+qKK0ZIggIX
```

```
lgSMTBDeggplVXtGYzVK9+e43kE+BgqFOCI+8Zq1O5H7wwYfIGdbP0WNBd1ApfdLgaEvoGVE90DO
eEIm0d1sEeg4ePOw3oYFolaEfhhmzrOLV5hmqIdfG5av5Tpe4pANaCF0lDGiEI77UlnMzURplhZk
ppR6RDdqiTAzvTx7IF+UG4OfnRv6vc/WGpq4fRaVEDgNr686BoBvEBuojkvVVwL2aw2EJNDgN0G3
gmXo7elE/8gJCdNKzvllKK4aSSo1dMgj/R3wHTzokPqAAUaFT04Oomu+z5gUAsUfHu1gaViegqMA
gWjEbTSpT7o6ONrvnlbG60POfuzNpApHTRdCh3g/xmaLkXhrjROti/XABrNivzKN6yF0093ULd9D
3gEEqRWC8GPUrCQhnCEkRdhZ+fqsvqmrxm8GY766Ih8TPAm3Qz2EZeJZUlXMaQvehiedaC0z9pDa
8fmxyIlDYWNZx/23t7MqnP/2pgHoavVAL6FlQxFfFBxxbSk1hAz2WV8/IkaMXKTvHkygI1ORam4y
9CLXkfnFGeRAl87PxPE0y8vlapJPoQq+ons/GxoYP8WohnLaiv0hlVIhJflbGXSI0xSaUBsQ49rV
tExIf8AgyyULdWHxVgDk8ucMSNLcQtiMq6BdKMFHQ0eROPcWwaLiw0yiYWRi1UeQVTWOsHV/zj8M
ozUFFJUKCF2XK4sdhAGwLM29qcWjER0qV1JueSgklEDuuiOUqPR/e59ZDD32i+LPa1f8jwAAZs7U
sKZQagwAAAABJRU5ErkJggg==',
        height: 150, width: 180, top: 62, left: 50}],
        image2:
[{src:'
AGXRFWHRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAADF0RVh0QUxUVGFnAEJ1c2luZXN
zIGNvbmNlcHQgd2l0aCB5b3VuZyBidXNpbmVzcyBnaXJsIOHbh/oAAAA6aVRYdERlc2NyaXB0aW9
uAAAAAABCdXNpbmVzcyBjb25jZXB0IHdpdGggeW91bmcgYnVzaW5lc3MgZ2lybCCprqv/AADyDUl
EQVR42uy9CbBseV4W+Dv7kvty97e/qq7qquqqprt6pWwFAQNxYVHEcUHEGZUhcEGdcTBicNwGBQ2
JAMNw0HFGHRxHR3AmJrBRERVZZG16ofZXb7nv3ps398yznzPf9zt5X5VNd9MNJVR35anIuvfdzDx
5zsnz/X/fb/t+RlVVst1e/+07/vjvkcD3xPc88VxHbMsSyzTFMEQfn3LDd8LvpSpLKctCijyXoij
097Iq6+ek1Jca+M3ADg3DFIufYTti6ucYeK3I1/wP362v+2ff+WeNsD0w2sN9Y+fousmfre7AcP3
QsB33UxxKvf/P9LlPsBV45Hzb9u745W3GFqyv//YX/puvFBcA9V8DVse28bDEVGAZnxqwrxNYv/r
Pfvf2y/gc2sztJXh9t2/52i+THMDK8kLSNMMjlTTLJce/8wLguwDir8Ia+b1/+Ru2X8jn0GZvL8H
rs33zV3+BmJat1s3YWM8LK0r79+pmKSU2N3bx02eR220L1u32K97++Fc+IwaoJzFpbJyzX+xwfNy
F/zUG7A//P/+HsVoupMwyUGdTMvx0bBeWH0dvVvKbf/c3bv2jrc/6OQTSr3gvPcYacOonmrCuDCT
BwsI/tWBpbfiqruOoD+t5LvxXPPBvx9lY2E/kw/4yfdav/Qt//xcd4z/+tj9m2I2ukZeladuWuZ5
H1nRyZmN/FnxbyzBKC6sLXF2zSuOodH0vdxwPTxdg7FXh+mFuu045PLhcXnv86eqxpz+wvWG2YP3
s2v7EV75PLq7d6wxYEzsuP02wmgBreQFWgM8A6nz8tL3OrpGliQ2/OXAdr5FmSS9P0n5epG340YF
hOW5VZJ5RlR5e74C+VzjOHB8S5XmS4fNSYHmFz5o2291zz/Ompu8vzDyPWnsH6dW3va84vP5YGYT
N7Q20pcFvYKB+1Xs1/8L/NH0hleYjCC5y2/KCB+e/iAR/whv7l6DEfKrc/B7g4W/2N9w85+AYRth
1gIN5i+16PTdoNeP1wjTzwmoEoZ2ny6u+EzZL29hLkmq31eyYludYeZZ6VpE7hu1bplFgsakKw7R
L27QK27UKxwnWRZbOKqM8sQz7xAmap14YnhiVee/sF37y+PylD58NLz202rvxWBq0BhVXju3dsQX
rG2L7Y7/96Y313Dinnz5gf8mbGICl14s3G6WhO68ugOrgcYDHU3g8aTtuC4/uejW75fnBfhKtRji
eKY6hCWPZcBx315AyDH3PhMmFpZTA9hoNvKZVVV4Xdng3jlaNcgHDCV4sZW6k6bla7otgWJalBX+
zHLNwba+A1Y3wWAG0sVUZM8dzT3q7hy/4zc7z8+MXP7Y4u/1y7+ihs53Lb4nwtwL72YJ2S4N/7bb
/9svervlLzWE+AOwm0lvJZ0KJDVDi6lNTYl0BngQbftJyvHdjXwP8uw0aXGA1sLM8nwAPPgA2w/E
0yiJfY99NfHhiO07btMwGcG7hmPDTiPG+LqzoQbReDZk+siyHSwrOxTaSaGEkSSxJkugjx1IBUOI
4XQldTyzXEteyCs/zxfE9MasqxTGmllnOzUrO8fm3mr3hh9s7uz/T3b304db+zbtH1x9bma5XyLY
AYgvWX+3tj37pk6+mY14fwH4iHxY4s55wXPd3+u3eM1WeH5VFFhpVkRvwH7HPUVHkZ/BV3aIoI4A
1gC33jLL0+beyrHx8qg0Azg3TcG3LflsSpxJna2Epk+mEEjSajCXJYnwqebYCEBv6nB634ch0eiK
rZSR5GkuaArjwlXmsXhiIa5vSCJo4uUxZuYUDa4XdwvL8pRf4U9OoXvYbzZ/t9Pd+bOfKwz+z/+i
77vb3Lq9xjcrtHbQF66/K9ke+9KlX4fdfBrBdAPaZVrvzRe1W53HLMgamaTTWi+lb4tVcBnuXfrR
I4z72d6cqMpjEvAEQtRipTZN4ty6wyPssvjAAxHW0hHW2xbBMKfJUHNfHZzuyWpzJer4Q12tI0O6
J6zfExLGQsCZ4j+W6m4INgHmxkPl8rsdY5LGea7PVAWA9rAalmFhkDFjoClw/Xk4kS9IiaIRRs9M
dtdrd5xuN1o93d/Z++NJj7/rQ4VufHjlumG2p8Ras/8W2b/wtnyflxur8Z/B7fQBrALA+wPAkAPv
rhzs7TwWB/5Yiid7mN1sT7KAJq9oM/eAEFmuJ3cGqZlaWRLt5lrtFWZhplu2WWS5JFuMTSgVkVRk
SRwt8nitxvIR1XEoB6+o2OhLiYduOpFkk0XKmVtNxPAmb3bpMEQsIGLAC17ADUGELIIwkjROcRyE
ZfqZFrmyghfcEniM2WEEA0JdYFKbnZzKfnRZ+0F1euXb9uNVq/bzjOP/m4KFH/93D7/3NLzYH+2t
TtoDdgvV13v7kl79LCnh1haZOXlfAGgCsBcC+BYD9QKPZeqTfbb0VH/JEnmV7ZVnargeglLk0m6H
02r3nRYosL0tgcHKzyqt2imNaLKcKNm4uLGcCYFbqg7r4KFOydCXYEXxgV5aLCajtFF802CisbSP
swqo6ODAXPirAmKwUtPxcLBTi2D6Ad1+iZC2O18TC4sp6DQCDJqdxtLHcrhgAa2Ab0sF7Oh1Yalj
mHKA9v38Pi0SctvuD2aVLV55rhM1/6/ruv3zsA7/1Zw8eeXqGs9/S4i1Yf+Xbt3zN5wv8wRqoFz9
fH8AaG8A+DMA+47jezX63exX0dw/s8EpVmc3Van7IKiITr97ZPZThcPCybVunwOq7l4spALNWMFQ
l6C2AR7BmWaqfzd81Co2PybMEVrQlzKOYVX2cJoPKwEiB1yfrmR5Tgf/gjTLYpK/ViLZpa2CJ52J
7nsxgLfOi0oosPk/rzY+ZTkcyX8zEDzrSDFoCViCddijddhN+cUum4zOZnB8X8J+jw6OrrwyH/R8
Jgsa/OHzosR+7+Z7fNGp0d4vt3bYF63+x7Rt+89t/OYC9uL4Bfv1toJpPgB7uNnz/IcBjP0riS1E
UtSpawPlYOr1d8axKcIPP+r1OJytSmY9HoLQJqGahJRDT+RIWMQNQEz2SLE/EBYV1/RA+Z67AZfG
```

```
ElKkGj8J2H5QVIMR7aYFb7aHSXu4viWcwth4AGWPfLvYZaeMBn4efKT72yX/HaSRZYer5a0oKq0C
WVXJycoLncqXX9G97va60m77s7++LBQs/A2hn5/ej3nDn5Nr1h3+s2Wx+f6e/80MPvfeLT/duPpV
v76otWH+1AWtsAFt9HGCNDWAfwnt+G17aanc6R6HnPYp/90F5W/PF4nJh2BKtFqCtvgRhS8LAk0G
vLU3fA4AXalFp8WgJ79w9kcl8Ju1Wb9MOZ+AnQFRkAIupAMwBcIKVH+8BRPB9cRgAIADWHV4FumO
cQ6HPwadUS9rqDSVaz6XKM1nFK0mjVNM4Ftv8YDk9gHa5XMpssYZ1JZuu2/BSfN7p+Vwi+MXM0/L
awB3HiYZy48plabUa2O9S5ufHaavdHR0eHv5Et7fzT8NW+K9uPv0lJ5cff+/Wwn4G27Yo4jPYTFg
nTVAqYDdlEJvVrixrV8zcvBAwBZ+U9wO8v8F3fRiVxk3LMHp4j1kUuR9FqyEsagJoW5Vh2Y1mh1F
VmWdrWKieLNapjGdzseFPVqDkL7z4vEyma/HDhqxhaWm9CURGZU32ysJ6ZqDJhLUNRAUNX4NatMa
O01RgzkbHWJ1z9UUTI5EGKKuJfa2WEwmbTezHhf85kMSPQJcjIf2O4T9n8FOb3YF4oMbn5+eSV6k
4Bii0WUoDvmscrTS45fiBWvaz0UTms5XsDVtysHeIReKyOx3dHb4QJe/a3V3J0cFh8fJP/9C/xvJ
ydvT4+7eA/TQ361u/9Vu3V+HT3H7ge//OxxOTj/9lU9ckLfz/t4Aavr8RNgadRvgILFELz/lVXvR
Ly15bdrDO8qKRJ4lPH1TyutvFoWpDZcr52bFSUBP/Pj07lQR0M83remH6juv1SvtkU1i0FazXZDL
WnlkFd1Xq4sH64bLINdCUwGKa+NloAYzribKDEgBbRQv4wKDzFReiDPuHlcYZhM1AugBoDj83y5a
SgwoHAOtwZ1/WyzmOZw3wNsX1TLABX0Dp9RgYkOLfC6xhq9VKMlh5tbbdHavIM286nfSqKm/6rjt
N56MTXLG4s3dle3NtafDrv/2x3/bOTxZ0slgAgMcRgPlb8Nhttdu7zdC/ahi2j5eDzxp7WZYP1+v
FYZGXLsBoGY6v/meOF7QaTQlcT0G6nJ1Ld/dQ0iSXk/uvaCoFUNbIrB80AL61Bn94DHUPraW+ZwE
Q2aCvgefLoNNWi2rC2jLwJEYurWZPWNfIHG9ZbkANWmwA1My7AkT6nka3LS7pMmgwfebZYgYKXah
vGsC3Ho/HAnZQr1LwezMsMHeOT+XkbCQVfOjAo8/rwvrm0u/0pRV6ekx5uk5NsU6Gg/YHr924/g+
7vcGPX3nymdXR2z5/eyNuafDrTEXMC3GN8uMpMe/6ffzldwEAYaPRGPqusxOnxcjz7MtFmb91tY4
OAcRGkeU1bQZFJbja3R21YHiP+oJMj4TtjqZDlvNzBaPrteR8fKrUlbSTVcT8qdFbjQIzCJRKp7c
naTyXGL7uOQ6mA7/R9+h7NtXKLgC6MAxVbobpmgyfXxSM/gY4lrpqKYVfm8CSko4b+N2xKtnr70i
UxqDgkUQ4phZ8ax7vdHJeR6Gx751hTwIsJHfuHuN4SpmszuoySsMVVkrS2lpV5Zb5eu98Ur7ffP7
ZcXn9+tj62I9/BGBNt3fXlga/rtsP/pPvkU/MhY09/O+r+Zvv+7uOY/sAcel57qNJFL01jvM+ANC
4yLk24RsqpdUUC7y3klFd/B2+Y7RaamSWOc8YFrSCVbQ9fxN5tvGw1F9uNLsAV6T1xB7AZsJHnc9
GAGILvm2LKVeluFwImKKhNeXBLgC2uijC0bQLBdOyeIH9F+L6vu6/yBP8LdWOBM9tqL/K57hYpfE
SFjfUKLDr2njUfrUJ/zYIfZx/ILMpjqPRldlkhsVnhYUh1AWghfPDQmXFaYK1zA3i9WplFMnteHx
vuf/IO7fW9VPFTLaX4DO3rIzA/s3v/0n9nWADiPoAwW+nV+F63iGoboK/Aa/WU/PF7PHlejUAZiw
CjcXwrBqajO7JClSXgPEVILZaVUZaYeJAhwHU1VgMgMwAmGwgjxRUtCenhLWzBRZKmrBuUjF9IvA
xuwCkp5RXLSYseAZfMgZYF6xagvH3YJkH+9ckwd/Ox2daWsjzIMBYmRSC9nY7O9Lt7eixcbE4OXl
JIvi8OX53rVz2Di5LujwRE5/fagSaY93f35VG6Oqx9LsNuXzpCDdXLgdHV4XiE6enI8ngc5+N6+o
pnE8wGk8ejtLqN5yfjT/v3gs/3/zwB/+3rcjNFqyvI1gtU2/uP/1V76mBC5QBsF9cJ3AkSJN00et
2b7q29b7VYvYW+KuF5YXRajXz+YrAb6lYGi3rHm7kdrsrjmtJFi2k2epJksxhcWMt5dPCDACPVhF
GGv/OtDAizzNN0di0lNwnLFuRrtQX7Q96sFzsjINFLbM6pQKAM0DFet/5ZILXptLu7en5nJ+fAYz
Hmi+lz8p64tX8RIHugdI220NYy45Mxwx44bj8JkA7kaPLbwGdhr+9nkq5nomVL+XK9Ruy0+tisYk
BYk86sNpSrGR3fwcgTmS5juHrTmWygF/tNyy46c3T0emTk8XyC0ej8Y3pyS1ne4dtwfq6W9bNTwM
/H8GjBatY4uHt7e48ZVTp+wGOHdNtL9brrLc4u9egj0jL1+52xGbUF8j2HBsUM5A5LJwJi7pcTTW
gVBq2WtiCZJdRWoDTBCD5ZeVFBCDC/4OPKNpHa4np1FaZhfX0c334kzn8T9Jo+qrsoqGFzmDRmM5
Zwm9lPW+I58JWV/Oko9EIf69zuibelyWrOgDFdBDMtt8aqC8bw0LbOGZ+TqsZysH+ZSwapqaL0sV
YBv2uXIVVbcJPHmLhaDZ8jTC3sThpmkmPo5LpdIr9hG6eF8PFYv2e2Wz2ntnopPcz3/dd23tyG2B
6vSyr8do1zpNasSEBcm7u7u68t8jza6kVpiwfXC5OqXMkw+Gu9Pq7WltrghKGfkMKAIwVQvPFuaR
5KXGylnJcaMEDG1TyjZRLFK01aBMBCKX6uDYAymzqWtvYGHwimJgj1aAX9kVaDV8Q1rSmzozoMuh
U4vcIr3MdS5/nfllbrJHiypOT0Zkso6XsDAZiOQFAxYWhUl83B6BD+KCMHGdxLGtjBgD2lJIfXLo
hy+mkjmLjfIBCgLYt3jLHu5kKskHFM5xjifOJcMxN/FvE1eon+K1Jcm26NJ4Zn00+4gfHrIeMt3f
a1rL+irf/8X//YQXsX/re/0DL2gdYdnaGw19/sL/zO6qyupFk+e7o+N7NOI4t3wvl6pXrWuubrOa
wVmtxgFZSRM9hOYRoEIftbKzzJRgvShj5WlqhNIkfiKeZWr/IdEpdNE8raRs1Fa6KBI+CuThx4Ue
3Ol1hVfBysdDXeR67YypN/ZCGF0Wlhfw2uTuOkyBkiWEUl3I+mW3SOqKWvTJ9pb8RfOgsjeDL0vI
mspid1nHx6Fw6nZb0NRqM9Quv8WEGWqElV4+O5GC3I038odPpaFXTejmV2XQs08WcdN1yw1Yb1v1
x0OOn79+51f+Jf/Rt2/tyC9Zf2fZXv/4LHwD2z/++D/gA68MH+7u/CZbqPVlW7M/ns4dnk3GPlrE
denLlyhVtJcvjpVo4lhKyRpe+JYM9k/GJRm/XALJQYQVOXJwstKuFlJOlegboZRLF2qNKn7Iq6+J
92xQtfuDrWVThAIxr0Gh+TgngMmdL2kkgx/EKzwdi+a4CEJYMtBrWF1Yuy0u1yGyA195VQHwF33K
Gz9a2O3w+a47T9VwaraFY2P96ea5VU2zR47GbTlM7hZgi2t0Fixj04Yu3pRU44uJ4+5227OFvgWu
oK8A8Ma1wkqRa67xYzd20KI7OxpN3p2n58Pj0xN3ebVuw/rK37/yGL5U/8z3/Wn//K1/3Ba7nem/
b39v/cgDsnWleHM1m00G0XuCmt+TatWtycHAAwDQevJ+plX5/qBlZ3uCL6X34f+cShKCLoMVZtFY
/tMzqjp9sY1ELbYWrGwW0YZ2+Kah0o9XZFEPgNbTCoMeknHVvKkx2nkjQ9LWcSi221gN7ulAkOE7
Saz/saIqFBRWay7Vcpb4k+kmcawUSg1UMUFXwjdkr69qhOH5HUzukygYWmDSa43cbdBrMwa4A2H2
xyhjn5klDiyEsCUNbdvs9acHC7sOX7cPKRlhcuEAtVpFlWl5ntpw9MltM37GK570f+0f/8zYyvAX
rZ779nT/5lfJN3/3/6e9//Q9/id/p9J4cDAZfWVbF+7I0vz6dnPdYJrizeyAPPfyolt95jqtAane
```

```
H8FFDWKYzaTRbkqdLreltNdrS7+3BR22o37ee161rjPISCARfLT9a1pVJmxJCg5VHyZT6LRr8cew
A+14qCMOgA0DNdT8FLJrv2uJYbIWrFPyu7Wi+NMBnP7DioLTLZW2RHb9+nlHnkoGodQS/t1LLyhT
OOkqURjM6fdFtxMoq3kbRfAT/t133L5SRHF66CV/XUMreCh3ptppYLEzpt1uytwNLO2zLpaNrGmz
LsHhF0QpExT+YLJZvn57Prs7Pj7eR4Y+Pl2yLIj719g/+3O+Rr/u2/1N//1vf9Fv9Xn/nba5lfAV
cvvclaf7IOlr3tX8UgDo8uqw0MsbN32x1hWWI09O70gYNvHTpGqyuV4tyVxmAY0kXljYHYFgEUYL
2aqURgM1qJDq0EcACk6X/dgGqIltpdw6rlkqdp5OKerKM8rJIooXjoLK+G2iLHMHu+dRNqi2rBT+
ZllkpLBYE+q5lmuvrqzLR6LRNJQtQbpZAstiB72U/rBe24K8utbMn057ZDsBaSgg/VyuxcB4sklA
1DCuQPBpJu7+nPrQWbWDhiJbzTQlJKb3uAD8zbaUrWfwRLUxcMytZryosdq84jvXC8vazydGTn7+
9CbfR4E9v+71/8R/qz//1v/9dTnewew333XviuHzvfHz/3XEW+a3mjtbiHu4fKHjGs3Peu6yPgK+
51nrc7rAvnTDUdIrZYAvcNaWdBQC632vLE489oftgbnOxmMtqvpD7o/satWWEtgTwqDhIXzRwWps
gUT01jlvlsdIp1qAS87FpwvRKqCJnGfOnKnJagsiyaw9gNgFOy1WLyU6ZKF4Ak7B+YAEl/FZaV2N
Feh2JHe7AAi/1PBqNASjvTAsqxuN70u0MxTbWdf+r31GaHJod+LkAftiVdDXSUkq228GnV2Deuv1
KzQCSlRZ0kJFMF7eVNURJ4vquu49zfrwRuv8hnp+TJmy7crZg/fS3f/zn/4DV6Q13AYtn1ovlV5U
ij4S+7/d2Ds0cN3prMICFSmSEGzgM++pDugDQ5OxlndFaRCvxhwditWrqulqOJVvPtBZXaWMMSgo
gsmPFx79bu33ZGbS0k2W2vCm3jo9lMVtK6nZruuv76lOm9FeLVP1S5l9Ju72gKYvxfQVEFDNA5ai
vmsSJRpwpKcp+V1YnZWnGXJQUMSPBPZme34NfvS9pOteyx/kik8VkJL0dnHq+VDre7OxKvBiBqhv
ah0u/uglqa+Fn0OiBzp6JX3XBENoK4Hw91pY8duIwp8zzvX9yJgVWNB9+93w2keFwKNPZAoxkYQW
DYXu1vP/oeNK+vthb3MLlj7Z34JYGf1rb93/7NxpNVjKIvD9Pkq+Ab/XuqsgHYautKvad4b4kq2m
d5wQwKlDWVqevhQxsTXOD+ib1bEN9QUZVKa+SUv8oXamiYKFpm0rrfdUPBHAZifWxz8A1ZdhlrW9
d6GBiP74Ha5izdthTqwWzqzQ5g6UOQHsplMb9NkBdsXOYplJb56j4EAT4WxmryiKlW1jPywDXfHY
KP3pAtWHxLBvEtxQX709BsaMlALV7BFDP626aZhes4RznFWoHkkmKzRm0LFsMezjPWGm3H8A3Nip
NHZGm8zpYDpsScC2w4qVgAzYL/UG94zjVCHhVFWaz0SmTJHplt9d5dvL8T64vv+MLtzfi1rL+0lu
z1XYBprcUafzFuIHe6YeNNu52uzs4BO1b10Xt7F6B/2ema9nZu6wpluPbL8r4/EzpbdzuAR91+sY
B2Gz1BW3QSQHQz1XxwTHr4obCppvqiJXWBsWDjxrCOh/2G7LTe1juHZ/I6flYPHa6aP9rHYcpcrY
NlBqwaTb68HenYpkt7bAhoFnc0AAop+M7sGRHwpCyRRUJALOApQ1YoIFFx7AH0sTikIH6epahVrn
AcZ2fncnuYCgr7DcoAmk0Byop4zuNmsIusGCVGawlzrVYa/UVa5s9HAsDarTmebGSNhY3A9fH8Vr
y0ksvYVHAouRZsM4N7ZHFAgdWbQ+wMD0aRfFeGi3ZPLSVgNlGgz/19u/+7rdaZVHtwgK8z3SDx1v
dgeu6tj3YBSBxI1NBQYXGDEtmJ3fVutDfm5/e0RyppmJwI7MA4PjkWEYAL4sRytLWwcr0M9lzXpm
u5OVGlKyytfTQrFwxAKScur4AOqNBTVi665f25cbVa7Lb7wL4AuuL250BHMq7ACxZGitAXbtum2O
FksWcbF5bfMq/0MfMizrdo2AvGDAaaIM5858RKLYbdNU6skifvjVVFVcpfXAX+Pa1zNED4PKyTi0
5bhNWsiGz+7cAxB6etbTQosyWak3r6LEly9EtabR3sAC5cnh4pMfDa1ZqvfIuP8vKyqLhuY1rUZJ
dTvNkGxXe0uBPvf3Q3/kW3KtZiBX+KdsPnwGtfR/u6v2wPbBoUShysF7OcJ+CZsK39BsN6Q53tJ2
NdbVs+CYoaHVxx6qvWiYLaXd3AWpPlrOxRmtJadmpUjFC7NbC2xX8UEaDKx15YyowDAIsY+CmI2H
Tlb3dXem32xT41s4f2aRSCEwtPwTQM3xeo9nXxvJSVffhV3oNrURqBD3sPK07ePC3OJqpBlROQOo
nZlq2yKCWadf+bVFpg5+kRarpKYuCTGZVS6Di+NlNxIBVkS718x2/qefDXj02BTBn6+Jva1j3xuB
IF49oNlKzOZuvsNiwwmoiDfi+eR4nzSB8oddp/8L81k/HB098YEuDt7D8JJTDYGKiGgBwbzOM6hH
cuIUTNM3leCRT3GBZtJLOYE+6/V1JFmPNpzLAUwKonpWAzu1rGSH7PFfrmQ4r5qMymM55UTtg2I7
G4cX0WVlUQEvGG9+m5bJq0qPWiZHcylI2mOeg3tEU/uRA+j34ss23ygvPPytRZsq9u7fUr4WZU+u
aJ2O8BZ9JuqxSpGvQzT2JpvckCxvSZs1wttI+2pzNCWALFRYF9p1SBkZ9TNBZ5kYtLB7z1Qj78GT
YGWChwvuA0XZ7X7J4IjEWjNOTl2Q42NfPzrKF7s8N+oKLBdBb6hsTsB4WkPNbH5b+lcfk+lufkOc
/9hHZGXbl5VduA8BDysZYrWajm+f5tbxIe2VeTGQ7N2cL1k+0/TCtqpicW3pouP5lEt2qTI+mZ7d
Nlsgl67UqNrBX04Jl6zaPJKesJ1MpoSeu01dKqGDF7R6uAq1aokaSaYe48QtYM//B5zlhW60hb24
tF6Q/CYpKkNJKMc3CIgfWA1M1v9m7XEuJAmgCAN68dl3u3HlZjIMDGVFkO1trjS4pMHOjKm7hhFh
IGNFNpNHb04WB0V2WCfJ1pMPT6ak0OvuymNwR12uqomG/11PR8H6nKys2xQPI7IFld41l+Spnyha
69WKEhQiAPbsne/tXVeupAAVeT08kAO11YVmpnkjr7wahNLHIze49K639h+TaI4+J9dyLSrHv3mE
AWFemRhyvr4HR7OE9rygN2IJ1u338lheZCYrXBkhv4g4cFFl0LY3XNkdIMLHfbIXyyFPv12BRjJu
xgE9ZALTMCKqWkRtq2Z5ZeUppHdDAFqwJO2lYoVR5rrTaXZVlWcEq86bXqqGCyg6WxOlC1QUrtXY
ugFOX+jGIZKYGwPKydHcOAFxDQUXN30uXLkk/LsU1czk95eAp9pQOJc7n4rsdjRBbsGyS134sI85
8HxuG4niqqZYGFo3l9LZ0B5dlMT0WXAVZwyp2QjYBrGQw7MkENo6tfBaOqVUm0m33tOKq1d2T1fw
c51DJ+dmJjtzY2bkMgK4kB+22cF28/r72yxp4Pd0BLkyrk+eksXdTDq4dyTF8/cMr11R+dXV+PzC
91qHjeJfKLP/ZLVi3Pusv2n7wu75ZG1dwt12pqvKpMo2/IE3iR1bLuZknicqlXHroCWGnXASg0tI
w1ykl/Tdb5tO5TEGVx7OVjEcjiaJakoXdK8yrmlQfZNsZKGae5QB6IrPpGXzYJQAylfunr8hysdQ
btqxYlG+oAv5kPJP55L7+fXI+kcnZbQAw097akgsC63RxUL7Ljh5P2/FqZYlMc5xskQv8jhgsQ/R
DTa8IfWb4mNrNA3DRz8w5fiMBRW/Wdcy06A4WDPqoPiNawmkAlQbQmo2OLgra/cNAEUCthSAslSS
1Zz1yk7XPLaXi68mxakmxg4eMgRSZc3hWk3viNQbS6e3IrWc/pMdH5RrPduJuq/Fss9f96PTlD0U
7j7xna1m326sbVn4OKA0Ny36oLLL3FEVJaQfTdnwp2ZgNkJ2fncJSxGpR6VMycprBxystR8awKob
taPQ2iemXdgHgcZ3W8HHDh03VLyKNpTi312hLGxaOBQZUC2w6PXwrjsqEcg6N9prCH9Q4DUDP3lK
```

```
n2dS8KkGasFQRVtL2PVi2iQStPbX8fN2Kwmt+VxeHjA5mmWnTOKVBQwB3MbsjVnioWk2MRPuUEGV
PKmi0Np1TlQ2WksfFjiFOklPNKFD72WwhZ6NjUN5L2M9YQc0hVQP8ezk/U/98PJ2o0uHuwRHO3wC
YezIf3ZIAVDvsHUgyP9V0U2f3uizP7wD8Q3n8qc+T2y/fxoKQWK1uv4HLcGTYfgsL5/mb3W/dgvX
jwVqUtmtWl/OseBvujSGoq4e/mVQ2oBq+dr9YBJ0hzXBHp6+lcSHz+VrzmXEC+pqzO0WkBb9NG8Y
NW8JuT8v3mOapGJ3FDR/Nx1pi6APQHlUkmvDrgJwTWNfZcgrAhWqJCQKTAmiGoX4tAcNCfsvwxQK
wFWwAS+/K4+rXchJcGYbSWIE+M5daMbhj1ur++MrzfIlzoAhaqEUcVDlU/zhj5RWO1epqlZLf6MO
qcvqdJ+MJFiGjlAasXq/LAg9DRjjn8ehcBoOBVGkpqZnK4vxYusMjTe/ce+Vn655YHMPO/uWa0mO
fMUCarmYS7lwVA59Z4Zj91hA8z9P6arKQE7uyinTRyAv7sErWg8JxXlGzvgXrduP2r7/rj5u+bbS
L0njIkPJGlucD+JVXqDDPcYeUXGGwB7e9tAZXZbaYSpIVWhbImzJPM62tDRxQQjvY1KzXhQtWXij
15bCoPGYDd6k+IgsraPGMwqpF0LCjHkAew2dlCxoDWQwqkUIzL2NWdbqECmmUUzGqlC02GpAq4rH
KuPCzSYHtg11VaeBQZZYejkHPs7LSQnv2oM4nnlYNcTFhrpQ1wD78bQK/hJXlv+nnUnyNqhKk9+Z
wV61zr0uAD7VOmOcfdNqS4/Wu3ZE5KHpn94pcfvgdcn73Rbl37442IFD1n40FJiwodY9HL/8cruN
lcUCFsTpItlyICd/56vWHJcDxj6fHbrvT3zMca2BuawK2YH3tBlA5pWEegpo9nqXpNfiqe2dnZyb
FtOmnsUghBe2NcFONP/LzSk9JCXWSG242Am13CBoautIE1XUpuULLyKQ/Bc4MqhI6msZwGG/mLBm
7vQkuZWo5Gahqcipbp6PC3/T7SP6yMlarmAGYNi2t5ACqpVFWy9iUKapYN8scAXqKm7mk4yvptFo
y7OzJ/nAg89VclqsIQL2n0qPhZuQFG9k52ZyiaHRNK1D2YrmqVSvyRJzcUd+X4uOdbl/CkJ9ryO2
7d+tp6bCQezt7ei5WEMh6fiINGcru4cNy/97L8uILH5a9gysS4pg6g0Pdrw3/dXF+W2ywiEZ/H4b
VkGQ50o6g3u7ex6Rc3mi2uy0snDt4uG/2INNWkX+z/fjf+xazMox+GsdfWJbV71rMJ+85vnf7YM0
Epu3ihkxkBCrYaO9q1wuT/fPJKSjgZfXXeAP3GUxxARptd/NUCJsbp7yxTYxBKdJYx7JUgqVky5v
T0NkxF9+DUVflSwJLPlutAdhCW9gMUMTVeqyUkoUMnueKVj7SJgKsthPUKhJqI0vtM2XNAief+1g
4WF1V98hmDyqlxqD1L734bD2Eym+pHx2BnjbCoYy1WL+2tgH8bAaSfOyryzJpnm+nJ45qQ03kfLK
UNRvTYfnb7b60mj78zy7oeaYW2XQ7Mp2d4W0R6PB1fHZcp6ncujmf/bhZPJPm7g0VdtPCYQ1mVSd
5vJo7Qetv2477vUWenWRJUrz79/9Pb8qbdhsN3mznL33IqYryGqzYF65n58/E6+V+nGYejJv2m06
mZ+qHcXIardgKFNh2murjkXI2fReWyRejTMT36Wey79TSQBBnx5A6WqwWMjn9HDcqfddkrflOBnx
sWGW2i9mb6XAui/h9T8WxTYDAAeVtwW918TMA+Brwb0V1maR+z6Yh3KEfWo/MUsUJVZvgwgBrxfp
cWnqCjH4uO3R8jsNo1ALhLMRPASjO1/HYR4v3mVqoUfe5spWOg5ctG0yBfbNS4L0dHYsBe6+VTmm
eaT8sFxDmkkn9S4AzdEJN50zPXhE76KjbYLCBgSr97L/F+WbRDMeaaJWUdhlazrOmZSc41JVVFLM
0T+dFGpd3fvoHy8vv+OJqa1nfhNuH/tnfMHBD9qL5+DcsZ2e/L43i9y6j9e50OjZXUaFaQTH8TAN
WMEqSjbVraMCFDeMBrEAXlpWWI2Rxg8tZpX06sXVOEzclQUupFWoE806tSoCiNLXqiQlWUlj9QjR
gRFmWXApOiqtADVmFBJCocj/2YWD/1UZfiQEpfi4tkg3ry7IivoZR5BT7zZIJFo+udgVRt4nuqQ8
r7FLT16x0wYkWc5nNZjLHApTCB9eRGCx4gN9MX5fRaVYwFZtxkT58bapQtKiI4QXwOUGP53OZzhd
15BrnwZpin6DH9Wm0O+KFbU3x8DpyFq1K1GDR8axQktWZVjY5Ya2eyEg6SxVtN7xjOt5HcRWWgO+
H42j1o0WevoTdn+BMxp/s+/x1f/Rvfk7e1FuflWTLDuxocb4Tr2ePx6vl9ThJw+XsPgxoKGk8kTR
h3etAhbMr3LhLUL/F7J4Wy+8OHgcIcu1h9VnID3+Nf3dwk+qQC1Y6sE3MLLS9jRS4pP9JORS7qoM
7paHFBOZG4V9LDGFB4SzjKSwMWqOba2SYmhDUAWbjeFmPJFdKnRVzSVXhIVMF/9Iotb7YJBgYoY6
WWnroclZONVcli0a3p4BvUAlxY/3Hk5EWbHBgc+HgbwYWGzawswOn2VHqz04jx2ppI8LFYKsu3sP
3c5YOqXMc2VphzACuy5pmCqo1ezrrlWWH1G3KQcUpLdMcXgcVnsh89IpUeaUpMHby+U0TV7EC7zY
Whlm9M4kWFx04lCo9/2Tf57/5zj/84Pcv+Ka/vaXBnyvbT/yDP29kq0UjW0+fSJPoC9I4emy5mA6
ADKOizCcsjKtN1XP4ZbAK8K/oDLqwToPhnlTxQikr1fw4gY39oc0G86C1yqDSQXC6OtVCsDK0m4t
VidYDs3PUMOuhyApW7f8Uje7Sz71QKbaV5laaKiJAL97Hv+mrKAau6vulpohYfEBw62JQlbUoOJU
KlQ6nGrVmeIzW1gIkWp09tYBS1or/jm2qnAsVEhn8Mksy1bpumWqJTKKEnaEUsOrarAC/u4UFIKS
GE64BrbCpDQV1940uYE49WUAL/kGRHRZLULA8r48t7B4qA0gYGeZEd8czwRhO8P4IO61wXOsiy45
xjjM8RpUylE/9ePE/fp/ceP9v/5y4V9/04fD1cmXG60Uny4rrWZocmZYH4+nDVQplAbpGxcLl7FS
pMGyErGFVOUpChcdg8dqc9gaKSwyytJB6TKwJpqVl8T1FypSulpkCVWtjqaPke3UEFzexwR5XtsU
xCAUwkGJfKBdSb4k3eT2NTTTaSxUKBowY0PF1Do7og9PemM5R6s1qKVhZTg5gSWKzvS/UMabPSiD
TAkfLscQsaFivtbGclvHSjcfk6rVr0mm2VC1Cj4MdRPRLAcxms62F+jxHVlRRPma92ugMYwHwYWG
PLt2Qg6Mr6iIwcFQlucAT0OsJPq/FHkxpMWLN862tc6VdQg786O7OVfWNp+f3gnR+/jSeCvHG0Hb
cPbzOxGOqWsqf5uOD3/51Wxr82b79y7/xjQZ8INdxnYMqi54qsnw/jtZ9FtKzbI5Bm5R5xGgBK9j
UOTSd4QEoJLWIllpf6wRdVQ9ktLTT6dd6Ryz1gz/GtA5vSP7NAehMo1TLSTNl5KV2uJAi09c0OFC
5tFVrSQXONBDlqy+nVJ2WV4NQjqaBNGrsbXSdYKEdUF4jtxTMuZFqiqdICu1VNVj0QB0lWlIeB4B
AkspAEv3vaL3S0kONNGPhaVKRsTWUU3bxAFh6rtlahcKrdCmDwQF83DONLDN63Mb1IH2dT0+kAYv
YAN3t9Q5gZSeSdTr1c+Pb0u4f4BquL6bDqwoFc9O05Frjya6czTwf9sp64QAXXlKrqnogDrersoq
xtPVfY2jeVEUSb2qwJvHKsGyHsgiflybpY1m8vCymC+YFywU/zfNMjapa/mXcWLgzTKZsArl762N
qha4fPaVApaKC2wy1/Y0N345Tz0wlUNhwbTtGTTcBFKZXdEaNY+lzqnZo1nUO5MZGVft5deCvzuF
yU8uleRlDRcB1vAYpLUsNQddzSVSQjJpK7PSpyrq9jn5jQR/XtjYpHVuS9Uzn4BAkDB5VGWi5TaW
JQheEfD7CMbnS37uswbAcYM5yAwsX6DUobKliZ/Br11MuU9oIYDFS7Xdq/5gCb2UNWo8qi0z19B7
TJoYcn12XY5pgNWPtkWW02rabqjDBgVyWSTnUqEgW5y07bFauF7wE+jsGzafso7PxW18dlPtpbD/
```

```
47V8vX/SnvmcL1s/WrchSJwyCm1m0fGe8Hj/qBf00TqJOVdTBxHKj35vCV6XgdQJrRTEx5jS7rZY
m+GHKNJdYaqQ003CrjZveIlpoW2n1zPqeolHl4Kc6n6qDhdXSsrWMqwJvUgWppkuMTYTY0hub3TG
09DrOQv3amlJzFCSBa7puPXiqdOt92AWYZq2ISKte5oZKrbDrJezs6LlZLiO+ZzI7H0tnANpqDET
GpwDkEACe46PX0tk/kssA28ntO1Iu4OdmCSj1QJbL87qtD/tm2x2vS1kFtcmjFhUVLqKxhO098Tk
SCKfYCFpquTl4i+5FA6CNlyuJCtZFT3RB8hs90Gz62+4CF7obr6bMWDzdbPUCnPOP4jvjbKGm1AU
Sq8/Euv6r7/hD8hu/+X/Z+qyfbds//wu/1wiw9Fdl8WQSzd4FKtoyDMvCDWNr/S6L3kHFVJU+KxV
keV5bPKZphv2BmkNPb/jZgxY3s6wpnrmZNmdrDLlUf7RSuIgOU6a4A2lwpb6qs6lAqhRctWSoVVt
R+oxmoT4cKSzHV7DDRfWcGNDCgkH/19rkZ2ldNW/Lfxu2ljAymOVrbtdUIbf52St16gd+ox8OpD2
4rjnZ1eguaLElq9lIpUaX05Ek0zNpDo5k9/CSzm9lb2uxnmvdMpvTudhQxJwC3+yNpWSpxgIm9wB
iQ3K4EKTt9KPdZqf2vbHwMBjHyDjnxbb7R+IGHY2Qa5M+pxOkScPz23Ne1/PTV2R0/6XHsXC+13Y
D+K3y5OZh6Nr2ix/mJ/o3gfqD3/6Htpb1sy4MDjPlut7VeDn+dY7d2DccB+5p0mIUlnNoguZAzk6
OtfaVRfekauwnZYtcWTDQcigerARVGzzcxMvxfWnCp7NdW4shaCxZ9seWtNqS2nUBgAabchUwYzS
3JJCBci4GlRYIsKqnelCMoNZ1ow8sam0Zoa10oTAo0MRUkGlq1FjB7pQqP2oYqRbO6yKCz+MefLf
Or0YsOZydaosbQc+8rQ5xNm1ViKDV98NdoVaZ6YA5mJGEvYH02LbHHC4WHSorUoaGPmYaL7QKi4B
lgMviItbc1+vEghLLOdBjyFZTYSTOwutYIklXo0rGGhOgbjFzqzTBDECDH+MEM4ddS77fl8XiDKC
993h/9yBO49WP44V35ZN34ZSvMUZ88KKuPvjX/mCdR9ta1s+e7Qf++jcYzU6flvQhyzRv2r5PQZb
QtOyi0RlqITxzmdQa4saulAxWIk7mevMHni0xJU74O/yy+dl9yZN6jqllbCwp0x4bn9GUGmwElKo
9sAhCEyfGBshWbekIcpUBrSlu3WXjPJhfQ5rsmM6rU9I5F9W1tLmdAHPdWgCNLqFGnO3635xhQ4v
rcn4rp58Hofa9EphqyZJav5izd3T4slCmtCt2gHMb35XT278g2XIqnZ0D6VNDGO9JVpMNe7A0GMU
gEhchughsJxStnKok6F8HO+DQq0wF2SR3tP6XqZ0yW2uqhsyCfm5B/SdWXZlGkqaxOxufBCsKhGt
PbktpMmjzTVy7x3Gcl/AIdBF99WF+3O8eHl+Dx1fjcRPvr3jtf+Db/sAWrJ81dMJ1TVCwrlGV13E
jNfElNslWbT+ccpJakqy0h5OWipFa3nQEb0AB73wtvW5b2r1DDcYYla2Wig3TLkDH+lmXtJZwJNB
4hQnGB7W/llby1Ms/fdaN1bxQAGRwRjIYznoWq4LV9h+AtpYz9ZSGcoAxUz4KSNBK1iBr7TEWGwa
MLKZy7LocUuNU8KmtjSRpww8kbLZU9ZAVVoxmh+0DHYHBAozRKz8h8fw+LOSuTpJj5RGVCju7e9q
sQEpOqVFbc7ksO2xpsClZnEjKWbO8NjjOJSj3cjmnLRYL4GexhsEod57VQTMVfLM1wKZFFFw2sxS
XxZ1yAWNUfjE5r0dUws/Ny6Jw/OCteZ55eHyBToF/9VFufpp4hHh8Gfz5g6LIf74GcG7qQll9dga
R35Q0OIRpLPMSTlI1MG3fTFbHAWjvzDCtov6yQdnKQikqKW0S1aWALNA/uHyoliCBZQ2GR3Ln9kf
xGk/9t0olU9qqNEgPlakdSq+oY6XFrkZd6MCHab4a9aU1pWO1ocMaSeYNbNWv571FC8S/aypHh1I
x7rTZBwDvwOqWVlkrI+qoCnw+wJpR+pQwKOvFgQoRVWZu1CLgl7d6qmgYA2TMoxI4HtgC5UKZ94w
Wx9osXsIvzzMXrMKEj9nHfnNZp6kCXXO+cAG40GmTAI4rWS1UBcJt7WL/C51CkFA+pjFUoBawzhw
ineiiaGmjgpSJDtDCQmCAwsdhe3C2nI93GFRbLsbamJBGy7btdHfwHbELp/gkBijE491MPeOi/Qg
u2xjX7jnTMEtDKYuUP/BtX6uL5pf8mb+3Besb2l8lDEyjaQZtZ3LnuT3fb0e2G+RVkTtUfGAZHSt
3ArcFKlxp4JEWzw99abMIXsO6lt5oVGfo9A9UYZ8dKYyO1qNuUr28dYVRVdf7lsZGtbCmykpz+Zs
KpFkacLqIGNOKMn+qVYNGrhaPFoGWuVTwFg8iv/Xg4xr4Fp/0PSw4te9tcsAyrHJGmsnihswSziO
vWCHkhRrQcpvtuqgDN28Uz6WaJSKtjgZ/fP+SjmbE8qWpG2NZSNjZkyYFzs9OtbTQLgNtDmBllt/
tCfNcJfZNuZqzk5dVD7i7c0noFCznM20EKBSwjCLneN0xLCarpkKxqFaRJK5ns2ExvAerveMEYbl
eUVYnFWcnfLnM0w4M7GVcyJ9+zddqbB7g6XKDri+uNkAqx1g9btVDqrmOGKWmzDQIaG0t6xt5e+H
f/mPDdT3T9Dv+7OzlXbJQ+HfwZbKQFGkRRbVlo2o8gaEq3EybZDLoXxG/2ZHp6QvSv/xWObv3ijZ
xU42+H7JYor+ZdWqoBq7CkvNNye5oNS37AVBlU0jI3KnmR3kDUXJUKSuBaqm3WzkANKg2szeM3jJ
QxQWAgSaxNrNXWQ5obL5K0HCmZzi0mXXFtKz0J+3S0b/nDOwUqVo37sJm/XM6lUazHmrlU0YUQNZ
ywTzU6QAdgI0WMVtPJZ6B5sLahr2ujoHMprFa18qqJ+Rli7k04fdz/AcHOGdYxCj2pt02bq1IQSU
L5nlZ/cRqJvq4vr+jvbS8lizzhHs5sDvepNHonGd52sJ34jIHi/OcZ9HZE47nD9MkvvMaV44jTq7
SquLKLihzhZXvF2A9T8FOSgtmFSyphEtgWpZd1lF2S37kb32TvP+PfucWrG/EjaLaie9Zhuu0cEN
2nKBp5FXmmEBFFkdmvDhXKusZjloiu3LVH/T9Pe3hJFCHw0tCpUNaYFojh+Mn2GBOsFEQjVVDhqm
NagQWg01i1D5pbQLqUsI6iFRplJdANtT22Gp9CVRaRHMTpNKxp0qftXZCfT/SQxOWkuWKzM2WBDy
opQmAFbpScEAyjgH+K9eckv6uznvFIlQEGrjJVUCtD+ue6oKh6hSdHa3/TUGNrfa+io3Hi7sS9i/
BcvbV2rMmmAuT22jIdHKiM2tIqzP4q+kaAAToudxY2iqIc7EDrZjSCiZSXlJfDtkKd2GIFwpUHen
h1NeRlq8s4j0vbL1oRYubuVW5zf7BiCXEYCFGmmUjsA/OGDnC43E8OGNzgccxrlKGS3UHj+hBCg0
AtR3137EWEqx2HS/A8z/xPf+dvOvrv20L1jfa5mi+r+/hjhl4btiqqpWbRUu3cppVtF6YQditbxa
hMHagEWDqENHy0EXOOPk7aMq9W8/JcrVWwHmBs9H9tTSXqpTrItKrcp+laiUxwHNxIxobCkYfUdM
wVS3kzQwKg0nqbpW5UmF11lhdZBl1DJn516qm15VTas2y0nsmel1+rqv6TLIpwq/yQn1CHhuj0VZ
u1YUU+PotgET7Xynnwjwso7TwLamKT+XDZDXDSVDjOJd4/JIErQNchIZKsXRbA4Cxqekl5qAFVrr
MchUOl0TqumICBftmWWOynmiuxWQ/LFaTOF3pkCymw7oNU9M8Eto6DpKBPVjizHHcQqjFxnpks+w
YkoMp986sQg6TaP3luNZt0H+KgN+nb4qfMa7ArbpoRFLLNMs6mm4qa8Ej5zU3zNqyig7NMreW9Q3
pr8IEuABpmkaHsE4D0EEXq76xWk4MtSy0lBTHzrKN5bM0qT/cvyInt35ehpcekXg5gzVhZwjcs3S
GG+1Q0y0X81KZgqFWUl0kUFtWVesn4KSuXjLMSj9LK5lIe/k7eKnjt+vUjwLa19dpIwDBT60mrWK
qO3NqSu3UZYQsWjfNV2vwSJXNWpmi9AAYBs5ot1MXBhc+uAe/kvTYZDyqqJUrOK+VKRJaY8qUBvB
```

```
bcTxJSskaV6ltQTOL81dKr3pUmbQHewDaWiL8nQtZtcpUFI4VUgRnvDyRdTSBjzysB2GBeazgf6Z
RrIuGAct8dEhh8Lb21urMOZ5rVbYc2zGcVv8+fNjDoiwd26ya+Iw78D0fij3fBBsa4dw58+N5POa
4LiMcYVJ3G9I/NbR5X6+xZW2KTOzNYmnK5kLKT/39Pyfv+Nq/uAXrG2kr1lOnSFb7yXp5JY/XLcP
k92+o9dLIJqd3m5aW0DEAU4zvyWDnkuQ6TsKXMsnk/Pxc1ilIMhxOduHkaUdTGLUF9TZlNemmKqn
UG7balB/WzMuswbnxby/yqUqTYX0vIr8XQCVT1kJ++HnsVqnn4ZSbm83agLfczL2pwfwgdKJBqKK
WUWX6Apa3curfKzuHDytKjRmX0hwrP99sKaVmAYQ2gmPRyMoIlpGR24WOv2AjPP1X5lDVesOvZwq
JWkyMAuuYj3gGF7olYaOL51oqas4AkuOz5nkt0zRTMTZmpY/v35Z+vyeNsA9XpTVJi6RXlYkDcFV
YGHCYjaWVRy3Hb05B8xdpkr1YZukHLdu6h1Pg4B5H+YwhMw2mG5RArpUgzU0w6cKaaupM02ebRfC
i1GlrWd84Wza+axpeowHf7cZydOsagNXI4admpIm0T9rhYj1Qbch1JGKpkp0UCguaOxqkiaJIIlg
GVjTlZlNTNxrkpbVRvZJ6tdY1gIXqvB/K6kGhw4X1u2g2tzbBIbXMZp2uqSv763QP7ydzE5RSoBK
crG4i5d5ULzHwVFVOXZ+j1Yp1+SKjviLWpk2v1KFY3JOVsm8Vn+/bsI0lLG6qwbRCLXyuc1cdVmg
V6aZez1RanTM9s5ro53KSAKPn5A6kv9QIztcLDXhRkTEIAGojr8/dqBeZ9eyuqv+bMOkBPrvVuYr
Fb6T9ustlhMu9kP7B1Z5Xpj8nRXaI98HJdOZk04Zj5oblLnAuge87O4b47y+ydIzzfYltEzjJOa7
dkzjVn2PUt6bCRj0fdhMB1tVSL5C5wedFK10pP/l3/4y88w/+1W1RxBtiqwpW+/Wj05cfhq91UOS
Zb2hDdKVUl2kEAi4DIIPWUCOVqrQAC6Fgx2sm4zPVG+IcmChNNGmvyvdS44sAJZ18YNj4d0aCK3k
QYKIVV7+J7XFibJrKLfWJNdJrlK/KvBh1OkdDTvRlqf5AjSPberXnVbt3ygeLgaaHHsjEeBrQ4Yk
z6qoVQxx+xc4ivJ9ibByKTKFyGz4qO4bUx2P6FwAk0TXLSosv2OVjOaE2t7PRvdjkiFkEwSNMQYM
tLGzMETNyzQ4kS8sSNz25Vt20v5yNpEgLFRVPV2fi2aWWE87mIxmDqcxGt9lFgdP1fgFIZ+/qK7C
M5ziupWFZ5/j38/F6+VyZx+cAc4XHNZz6Y+1u9ztw6l8E//QZ+KoBfpqmeUF9rRqoG6v6AKhlDVR
eL/78T9/zzVuwvhG2dDV3osndA7s1vA6z2auKjAV+tTA2CwIoew9fi/k+jm2kxWB3CJvBg2Zb86S
MAFPdkCkRzjItZVPkQMtF+lltanQZrKFqA/tKgeBchwnXFlGpK2EK0KglszbpGOPVyXGy8W0ZpdU
eV2tTLCEXtb/GpoDiouDfUYpcWxNbBbPVj97s01Cf1tIIMoPHnCzgUoFQywND+LCBWM2h2CHLDNv
wPUNN5XABYvjKN1xpcYAWq7qYgokjFeom06BvSr0oIBLAndfXAm9UVQqyE6Nu9aNb4Xd62rWzWtx
Xmh/gGNqNpuwMD3BFHClTjge5L9PR8RNFFl/GtWBKBuTdAPL9OyaVUdPoDnb+Ij7l3HbsHsC6xt8
9LCDHQavzVLPX/auwor+DBRAXIP3PqO8nAeobXfD/TUODy2hmxPdf9g1xLsPSXU7W5w03aJuctu1
ghTejRANCap1A0aL5QmJ229D3JKCzWOIkAS0sZJ1lNfjzrBYpq8wHVpD+IAsqsoxVNwkT/NoCp/N
ZqelrsNbYEduzazExvyFhYOicGQaQtJqnqjWK1T9VS7kZTGywY8XStrz6pOpmAAKQ0WRzo0pBQJu
bSqJiI/UiF/XJtJBcGFjRRGlS261raeGrsuhA9YupZggfk7XIGcd3lEwJZeJZAahuKQkspt891Kg
v38tIMBvrKTC+xPXM4rn68BRtmy7gix49hPMNJV+cSx7NsfDtAnd1/XWju4u/rcQFvXasfbXSnI1
DOhHNJ1cavR249UZYmu4txsKYyi3S+N/iNP8TTmeG6/pf4Zz/AwfYZdHyGdvzn4FRTQHTGzgRDyD
N1ZJ+mkDl1fvJv/unQIe/fQvWX7PAUpYaWR65levtpNGq5wVdl+CDDyTp/FxvcMqlUD+IFoFdN7e
f/Tnp9br1TR/neoPfvX9PC9zDZkPScaoBFQKKlk1FyipTq4cWi7nMpyuZw9dlAQIjqnrBqd/kuht
liELlYfb2j6Tfa9czavQbsdTX0xxhpQ5vTYMYxLJeHQReB6RMtayvzeOSHRDs1DZSP7wCtVd1wnL
DImJlBckqFi5WLJpgVRNHeVDLuK5HdnENQIulbsOr5V1YwuyrCBwLODy26LF2GrR/NS90X73hDbg
GMwDtvgSdIxnfuy1np3dxHYc630blW+KpNtBn6zks6G3ZufxWpeDtsNYuVqHwepFKce06YCVtq6r
alu9/KM/Sl/HUR3AoZ0Y9Q/pFLFgTfAcT/Px/qyK/BwB/FQA6Mhg6/wyBKhvdq5/6e39a3vF1f20
L1l8Tvm8rTbTz6f2G44d+kWXmGhRXUxFaLNHQsj/HaGkqY7qGhchSrUrKtGbWlFW0wO+gtUYqccQ
xh32NBmugiD3ioMkrajZNZ6rgzwJ36glHcQbgOHUkeNNR42yS/znuoLPxufrA7VZT2m08eqDdZV0
UUVmbtrhNa1lNKWvazJgK/Vc9BQaCrHoRqGhtseDw5qTVWyynMj4/0YL6LM42Cvy1vhMDQZybetH
MTqCqDGitvYJjn0m1yLS5ndVbKsBNOs+Rlgaj1lndZwsLuYTPORvf0hY4v7Wr7XI+3AfO/5mMzlR
Foj88lNXyVItKdKQGh3ndf1na3T3seq202wvaGlfLorEbeHuUhQSfzppFbj7uggYV8eqLcRl/BMc
8xaOlkXXTbJicf22Yr8DaPw+gzkg7NkBVL+BTAfXpr/8OtajGg1XwjceJ3zRgZaBFTzjsYeUeaeS
SlNDEjeuwXQzWhpTPCwgiU86Pj6XTHUiAGzdaTjRtMgaoKqvOwfL1vAmHgz2llARqDEsRR5EE7R6
otaeN6ObmOydAGIzibBkNBwF4lDdZUykwKxQY63VUS4nSysPSMv1TixJaWv1j6Y0GENp1xFmjm2a
lxQ2VWasYMkKslhT74XR1zmo9O76jn88SSgJdhcYtFibEEjY6YrVaSltZCxyo8qBX54CpNJEmao0
zlhbCj2eQqcB54DDE6ww1IMX0B9+rM1/LfVmOT8Vg4QRF0JtFPbAL17acEjj3pYdrljlM2UzFoMJ
G6cocIG92j3SqHlwWHTDNpvSySqhvnphu845ZlX5e5odus/snsvWsA2v6z2u/1BzVQLU8/B5Ztnc
dC+HLFSlwbVEvgArQ8iIWGvW9CCYZF/fIRfR9A9Q3mnV986RuGHENArDWaeS3d+LVclGyn9JrDnE
zTvH9reGjuSqqzWFRtJSNkGr4ddohA0WMVpHe1FNQ24sSwPV8ihv/+gOtJA45JqVjqoM9mnXbWy1
0RhBUm6iwSrPgdu0YO6DYsc47LUFNCZIM1nAVxeJpI3ctlKZ9POwPh/WkJbU3PnJ9Z+U1Fa7qaiW
NQQPl9EH54HN+GOB8OrCajoRhUwM7lFgR09ykGTdFFyzzS1M9N80TA4yUDbUBnrDKNUKe4dhYJ2x
SzrR0tTTSTNiQ34eVDdVqz05v47UL+KZt1QsuVguZTk8kqWJd3MJGQ7r9Piy7I/FqhuvI6fCnG9+
3rRVTBVwHvzG4C0qfGMwQW845jOW8SJMnwvbwK7Dg3QDbmIH6rgDUMY79Pq7Ju6QWP/8o9ZoqLdF
WoFYXQH1ti9wFQH8KVvUi5bYNMP3aY1VDq1WRjcxGOE3Wk9SkEK5KiSRq6fz2roKGIt4BLGynPdS
xjExrTE/va08lhbpJ+2gt2Cqnc2c26Q4H1sTSKWyFtqKZm44aWiZOoWMuMmGENKutKntWax0jQxc
Bgz4pc5m4p+Kkko3NFMdmKWKk2r+MZ+nN6JgbvaZMo8qkvkZhaqCqHqAFCLn1kKm9nR1p9XfFDdt
Ko3O2yVFZMJrreTBlxdwxj/HC+nr4qT26rqvjMQzX1FpfrQf2qI7YrvWEGVyyChWSq1ZTZSo+FoJ
```

qz4Yfe6w2ioUWdSugr504nlkrWLR0DOa81loGG+FwqhWodBIv1cKbQciWv6umFGe4NhOsJg2jMuE
qB7ej1fLzglb7JaA9xtfyfhzYXUPD6mZVlNV9nOhBXRtSucDobQD0pXd+3V+JXntPfBz1/YT3zRv
Jur5pwEopTbfVT6LR7XtZvDg2bW/tuEaYJCszBVgt3GwJ5UXEUm0jjkW0qaK/6QwpK53bpgEdU+q
0jMWCiaCxmQguGytVaDqDqzetMQM5HIl4PpvJ87fu4Yr7OieHNJE6u7v9rjQ9WzWSONApMFks4Gi
hfJaCmrIIn7pFVPtXtfBcXAZ5VEvNAuM1tSCiLqSoazLKqi6waLZDCVqPKcXmyEgOSo7iqUa1x+O
pTFcZHvAn4X+fTlb4fQGfeajiZd1GT9qhI/vDlhwOutLrgSJvNJeUxhulLnDqLzIgxfY9r4mFaSb
l4lwVEz2vpzlryt+sqW0GV2Q1WsiqAYs8t5SOU2fYtjK4EDNJVxNcS7wW7sT43qn0hpcFFwMLQZu
iTD3DwupjWsvKstZes/sxnPKTeV7832AyvSLNQHfMpNJibGPFUMDGoo7wXUy0/vETbK+N+hKYb2T
AvnkCTI1h5fWPUuvWs/eyZPmyaTmzslx3QRNddnwwwGQAmCVoLQM5QdjS+lQq4DMwo4OIKd4NKvj
avCrVIUhx9d9MgbCURoGaqXLfKycj+dkXbsv985kss7r3lGqDaTLSuTcn61z6oNt7vb7ch9VpOJX
02h1peJbO0Gk0HFV1COimMtBkMuq81uJ6y6S2kl3rL5Eos+MnzdQia5EOzsmExcT5qnWPAMrZfCn
nZyOZJiWIZUtuT8/lldFU7ty/q1bWdkbS6gzk/nImPhaC507Gcrk/kpv7Pbl5+Uha7bAu0ZNqowl
V9+WadbpXVfvDoCdxSqkbWE23peqhVLHIIk5m76obQSoewQ/mqI8sGkmjc6gtdfStvRyWPTjQNr2
05Mzb5KrfaCam46/xWbZRFZcME6uWac+9Rvd9+KJuFUYRmo53VOT5K9T1xvFRZ7jcBJNMpcCf5du
banwGywKj6X0jXZwP0tXiJozT7mJyauMGMdivSYuRAmQcdcHxF6SnOWjplEOblmtZrmOt/KG1JKA
J5H47kMduXAP162g+ldFd1gGw1/P4fCQfe+mu3F1VOiWcSon0h0uAPoW1FfigZ6DXZ5NzSWD5+8M
jVfnn0GLWHdN/TtOs7rKhZIuKdNe/q9CZjt2wVYSM6GQ6RT3hqu5zpTWl8gL/TT94Dd94PAKIDE+
S0pIf++iH5Wy5lNFkBJB54nd2dTrd+ehExmenssTxrtJCZnEuy3ilY0Dqzh/RCiadMMBJeWyWp1Y
ye1p10jqOO45qTWAd02HAVw7lfAIKHAxrFQuAyPFdHGciYetQZmfPw91oa3eM424KMlhJpQO9wCb
8Nj6pcECb9+HKdOGf+thHHy7HQWkUTwDI74Jb87DtNfbBSu4Cm9OK079qH1ULgO/99L+UYzwO3/G
bPuH9cfB5XyLHP/PBT/gc/87nt2B9nbcP/5O/LGcf+Xe/6DF46wekjFYl6JYTL0aXsjThCt3ygtD
kgGR2jKTrqQJAI7xM4UwnAB78O9OX+XymAR5aKlbjcC5Mv92Qt1y9WqvJV3W9YQS/dDJdyHN3T+R
+ZMgxLNdaA0auvP1d75VLV6+pn8wpaqMptY/6YIiW3L77CqypL11YWRYUZFoJZeC1pYRuoGLhRmX
UrV7UI9bmAaPOv2rS0d6kJURre9M83USINymc+Rx/t2SN/X3k1iuyd+NtMgUlH+4N5Nbdc4BpJe9
819Ogwg25ev0h6fYHcvvlF+FnLgVLmMxXaywut8XK19IMAx1t6Zj1PBubfrvOvMk0QJCkKx06RfK
m/rsTyv37r2AxGKivW3D+Da5hGHZlvTyHi3IAKrzQjidaWFJsxgZsir1xCkE6Y/GK6/ide5YuVJa
ZLEaNJF176XLuJMtpuJqOutF8dN1vDL4U7mu7ypNFpRUhFQ+kvPBN7wOwnwx4b2TAvgnKDY3NjWz
Ix77vrzE8kThh+5bX7v4sNX46vWHVDAKtJmKaRUXJcBNSwoSDqFgiuFonWmL4ajlUoSkdvr7hufA
rnbqMkK0eRSrL+UI+9vwL8FGP5bkXXpHFdCYt3IS/82t+n7z9bW+XPJqBZjvyyNveJbdHEzG9lvz
G3/o1oIigjM2hfPiFlznZDuC3YAFFVlEuk9VS0sLQelz6xDRbZX4hV1ppPW7ttHIWa1QPZlLFxAB
utgfgm1qnO40zuTcay1ve8QF5y+NP4TxMufHo22W9ShTsT3/+b5RL167K7l5fvuZr/2v58q/43dL
pBLLCQkVX4NLBVdm/fF2aoMq0pjoOcjOPh3NnqUXFwg+KuvFactExcNxUYAycpvrirHRyWIwSJ7D
ec7EbXZU9LUDxz8YnmnM26ZdXdfmkG7Q0yGds2mlgwmFM8yBo7WRh2Bc37GgqSjNZHHw9udcyDPP
LAFQGmE6lLtl+UPDwjjdgddKbG6xq5UpNR3BOqsD/EaobgB9anjtp7j30UViEE8v1clbqtlpNzVF
STYDC1AxHLOFzThdU4hetBaYwWFGkWjgA71XlPjng2Nx0vrBAgeV3vHn3Ll/R6qcWLFB3MFTa2mp
19IZrt3q6r3t3npXdjg/fMJeTu7fw945cfehRGR7ekOeff6meugYOy/LAM1YbFZVKouasRGJdcpV
tvkV3M1WirC2b1BTacpsK1oIR4Ggps2Umz334ozLcvy7Dgz1YtUCGg3258+LH5KmnHpYbV/rysZ/
7UTk/uS+Hlx+S45efl+d+6ofkUq8j73rrTfn8tz8qj968KoPesAYc6KoWRKhYuVFbV0bFnVovmXN
gLSoqcpIenm+EpMlp/fyFjhWLNDIG8jy9duzhHZ3dkzGYSVYyql2oH8vJ7FyQqiI5xPdqG14whtl
NQMsLFrzw/QStNitgIcWC1gZV/iJcjL3XAlXTNJ8kkHSxfapg0i/13i1YP9ON+rPwAQlOYW0rwap
ld6Xc+o//VxX0LieguDPXb65hbfIClsjzA1X3Y8mdYbhqRXJYI/qpSwpVa9N3rUnL3ylgxvhwu+G
rdakFuDZi2aDRTfx7Hzf5lcOBDJq+5hI/+P3/FFTwvuwcXgPNs9UK/bpnnpHLl/bl5Y9+SK5ee6t
+Rq+3oz4mBxxfjMpgSmcJS0TB8fxCIA1+7X8W5OTAKqlV/jUHixvY4AjGeCmG38bTplw5eFhTSqv
ZVINgj7ztvbowcSE5PDyU8emxUtdbH/k5+fff94/kJqzsFz7zbnn6sevy5KOPyP7eobTaA/j0jlp
W1YzaSKVe5H3pKugUdgC63RlKujxTK9lq91T5gmWQ7HEt87pjKYYlpY+u/cMZh331ZDEby/kZfOf
JQs7OzmS1XNfSNFStMINGEq0H8PubeWmYWbRSHznCd8VqrenpHfjct/uFEfwRg5IvBp0HMT4Ti/p
GKzX8HI8GG6+WpnzcNj9+VpLl2Q6++mG2HntO2NPxh5xJyim9VHRIcAcvNKiU1IGSslI93hL0WL/
5qm4ba/rBg3GNOQceEyNmBcD1lOreuHJJRqtMbu/uyC8896z8+x/4F9Ld3RW/2ZTlZCyz8bGY8Ka
uHF2R+PxY7tBvbYVYBALNh7bbhyrrSeudJ3WZIMHKiLOW/m3GGmrD+wa4Gk5hA6gTaK0v9ZAmsNR
j+JslrPDi/ESmd16UE3w+LVmERerkpI4GczRIv9sWZ3hJ3vOe90jbt+Xo6EA6WJQ4tJ21zIxAk2Y
bnOPIsJdtbWb4bLSOTVh3Ti1oYNERR5a0ikWdj1Y5VSobWpWEzb42n2cp/HaDBRKuSqAyF8t0EJZ
FyeAnUyo1Wq2kDR8abgsAnitNpiAA9ePEHyRVOfFWs1MsQLUvnFeuLKfnveH+we/P1zxR42M4wOm
nC8JfSwv65kzdXAiUbXzWi79Fs/vNdH76VGl6Dy9Obzut/beAap3oWArLriuIElYVYbXmsCfRrpt
UrarpmJKu6tw6VR5c36mjm/qqUksMm7AoLH6wzFCMKJHdFqjZXgOW9GHQu6n2clpVLIPQl7f//+y
9B7Rk51Umuk+uUznd3LlbLbWktmVbtixsbJID9ozNYAbDM7PG5sEzPDBhHm/AngcGFmEeMCzSEJ6
HgWEYYAwM4EV2toJlyZKVU6tz9823cjinTnr72/9fdVvGHmyjUbDukWrdvvdWnTp17r//nb79fYs
nqVopUb0xJ6Exeo+Y8EkOLkofViJ6iE4BcG+qdpEMgstmlMrYmyldiVgVmBCmA3QgSlck/877K7S

```
PQ+LKgavEIBE9KDDEROhUx902RdceE2Nz8yVBF/kcslarVRGJdvl9XczPGgrbDPpScA/LNBCq4pJ
ukAj9IG9E2Op4AFPweXiFtRGKs3fPg1ycP0uA+wg5zCQSNv9hb0tek3LemuMUAZ67zblrFmUymhg
AQcav6e5sC/yx1lzi63SEsWLU2+QNKPBQFEShbtBrU6e9KUEjZnXJKby5mjd+E3EHG6DJxpp+oZ7
12WawX5bGatj2kxysWvHTVBZjb0m12DjU3L7weAFV0vFwh50SVKUcgbv1+y1hgTBd1eR3eJFBhxQ
zngbv2JT2Zbd3+H3y7L2kuAMWQoypsZmk8MIIX4GG9x3RSl2eq9Bio8TGURKgOopAYKAAH3GhUBY
oIpgBMQAQg12BF/sIkDvgcBOFdFKFLJWPo7iEjoQo3mGIVhOwqUmcRDYOhNTCJMjXCoMrWjF/jTk
PBLIpIpvf0yh7lM6XyfY592aPZ/k5AWA4gu21tGI7zeCUsklFlqZysWUIQELlNBTCNCypNOyLEXq
gfDESvkcFGkwkZpEK8/b2JuUKJQq629RjI6409onBsnXRqLsj+OB6hQ2215GpHPwen2XYWeXoYiR
wzkYuL/fEL1UBcDmThqPr02hSQXGrXKlTfzSUKaB+d8duNg5+VTzuPcQXMHpOtx6/7IvB6RU2m8J
zulYaxXNB2GqMhx0b7HwYKAcA38mZIp6MiBIV4LEA7DXtB5sh5iy7rU2Vowq9ykiKTHJ+gCVSZSS
gDxWKUlE0dyVsFZ1UNky7VFDMEokv3keaoJkleRzYFYSVgX8fSq6dyMQPvH3Oq0g7iaYqddn0z5f
NKtSqiKKkMuB9wY6YTgfUOT+EopyJiV5JgS1NMp6pQpRUc/MK3OBxaA+qGduSyq7KQ7XSHXt7TB2
BhiXmUFciCgn/HYEeSr8X87AwctCPEm9Gvk99kIdneWGKBHwRAwGV+f20s3FWVPgwUDAe9aTIN0E
LpzovqvI9ft6ouyVgilrzsAiBjXvb1Oe/AfLbKOguFZorHcv1zpfqc9C2ubrb2uKNELO3rkA/Obz
mlId2vlCv+oWEyM9ETvvl6Vk/z58ERaAoGNrsFQ+yHdw06PYKHodcAeeriocHzBAjIfqesNeSCrA
ojaeCy0WKCI8KBXAjVagdAScIrxJYkhJZ/FNVN4RiMV+MbajwGsPjQpdiguZUFaKE/zdW4mYyMwo
qBIDvIZYcjqU9BHRVPsc5o2PthvN4LakBckD+xPNlirgbhoxwOhECmkQ8NUbcTNHQYQPNFzSpmxq
NQ1EMxgXjFGZG+WCuHulTX0UAGjmxoaUjcQmpKeeTvogewAcwRKqzqBLzPQKJucteEPIeE9GHHXH
OvkiXzz0qc7LALEPDJklc3gxVRR5q7jBQEK/lOYcO0bZiD4zNCkbocWgsd1v4pQjV+xOckgSWWzr
r8PXni0XpTY8RKQiizNhvXJGHPhuLR3t9Vu1Tp/BABSBwI6fceMQu1h4IOdZFqIRhc5T9IyiBW+C
tjQRiiLYElGIAYADUD4vU1syDYEjDPCcWpW0o9jyw4CNMtWUqRk3rwIv5rsdGyiEozmFqMjNcDaZ
bQE+qxZZtWfSJGB54nCxe3kVemD76wI7C0oqXnPV1lYcVjydUKpoHV8++ijgWe0KbPbcnAH9LyLg
BZIBnROvFY6+K/itaH5gvRZ4HCKDMtpqeyosxzofcfUo3o98TnwP8UsIJJdcDTPNAaF/AiAiNWNR
/wPGEanHEHhTgE6Ca5vcdp3Q0UCJbQnDOoT88Nf8tMOUDpTuT70HeKyrRLTcnYBVcm3jLKT7byct
XCMMmg/aJaLzJYXMzm1852EW+DaPmDRgk4MVpgAWj/ULy0WebUZtfrgY6zU9FFzSaCOM72OAn4Ti
djEdgw7vFL8+FyL2K9UUqVGpisOEk4hBYcSshQrVB/5mlws4PAwIYQpgM2csAojdt3OtYUcJQiaO
tKRGaIUYLgWNLS2Wg6YOfkaVG4GAEeB8h4kZ7CERj/F5Fzt1A+wK2f1SqhYhMyA8UZQul6vPFxq4
+E7ifDE2mBi+OCEA8v5GqsJw9NHJpjP6pIRVTwAxCI2OaStlO5F/ZAPieyb8hwYGSVYopm0RTp3o
CNZDNI1EhPB4IPywbCnc6vMHG4HvCyh9MoCTv06izrarbiSJec4tF2QSRX4P+BQYLInCE9kgdMCy
BnrNAFF2FH0b1XpgeDfTRMTWVFVPLiQu1/UhXLsVRUHFLFUwE2cP2ep3/QieResMA8TC+iJX0uR5
7xvoUHZBsDERgaiTsBPCMsUgMJmJc89e8cuz5pQ/kStVVw3QF4q1EihKp+mKZIRSTCjAa8o5MbIp
uSw6D5QDuQxoRxgJqThNeV4eThikGidwTYaKTKeIzIHgc9B5h0ML8oMm+NRNimgXilfhNxZCR18G
boMjj2sAq5xRiiFTICeU4spRWqwWQsakq3qamfZlGE2qeNtKe0hFqToDtbfFUpib7s5ReDjwVDBD
5L24KbwJS+YWXTtMZOyqqzvBsM0ZGS8uAiOqeq6VAHKVLa+q2juY6nkQjMtwa7aydpWrzEA26APw
HAp+Uc6UKVAKdW3hjcFcZOlqAqh1y+HDUESK7pN8TmhlDnpfQaOeMDV7n7uaF/d2dLeqsnpYlnkW
B7RRrr+CTeNM18qIv1Gsaxud+7BnrUwVeyv7BgpUZUF2O2Xj449nBV791s1Rt3GG4pUm/3aZOa4d
63TYbgS2D2GAqGA06KtQT5vlEVT7Z8LEg0ee0Dcy1RkJFlulqqRKWUgaL3BRe0JE8zpwx9pvsDYR
meHqdqKwCvYPwGV7RUZM88MaCj7VV+Hfln0w+I2apNbuh/Fs2nFTmW5UBKa8+LSQBeK/C81jaTUK
RSrGA89U5VEXbAOkCaHhBDJemeljekPcSmhShN7U1+bgr4TJYErOoK2gqU79GKQ8ojihFysYm7te
o017jD5+n1s4Fqi0clTlhDB04fA71dM3SiLFAvyiGC+9aLM9J3xi9WGCKIxDxS0spkQcYGzHwn0G
ao78juGLX8Sjs72CQ8BCH13P3/d4Pf1GW9qK3/5wmCn/yY89Ynypj/RyPf/Ac24tsI/mE63ldFDG
Ho7HWf4F+q6sgcbxj5/ySTK6QpbxhjH5hvijFGSz1PtBSyIXRnbdVCDxVBDdNleeiD4mwUnK/bKI
WseRdagMQYEMUi/E4Iipq6T6uYuxXxj8N7RPx/BJ+T0PfNNWFtUTlmMgtM/PJScEVGxi8PPihgMB
S1zXZVQSQ7C9VFV4bkYKMnCvvhs+EnBZkbCbNKFBRSU5FIsQXOKG8P9pKpAbkRbEOUQvasRjN42C
itXOZ8+cqe9hTlK80ZhVtR/JbtVkB9gkfDoMFyMLnDRQPEp1Z4J8tSsc9EdOy8yXKcRojg/ehIqe
rLB4UlfbG/FKQDFvVSXe74rn+F21pImL1WY89Y306DXrUy4w0fsh1nMex0MJA0WqimlguKmVwW8J
EW0R+HfGuprAfKhkMnwJp82SCGJoRjgEPq0CoZE49oKWY4E3NGqhgeakiNUOYKfqsJMz8KgpQeSM
8g4R4U9LCRAlbZUmiq9eJglHKQo+nu5B48ytBIcJ8qBFGwkU87cnairUQXhdGBuMVzwQeJ9OYFdI
SznWNaVFLRvIUOEL+bds691WE3+JphYw8FYOGt1fQSEPaRb32Onu7Al+KR6uXTrODbdCAfwYjRa6
uRg8dCY0xdijskXw9frHO7xhRqb4kXtT3KyrvBZRSpqSQ7+Zk88M9LlUqlPFGGo934vlDJz7sVZp
/VF46cikOR1/0XOvJt/20kuG84rFnrE/j8djf/AZHn8PznkOfLFbqbKSOmhZxLYH6FX2L5mo1qhZ
yVCmVKZdzBeSfJqoPC2QQpkN6Q632DfZEw9EppbObL6JqKr3URBaieijvg16oGKSR6YhAeSQBV0B
QSjiLUzFy5HvKUFW7Bbmtei9DwmhhpxD2w3i2gaQzQEgyA0rMWjAShifyemH7l0LXFcWyLNHhr14
kspso8IeEy5Lou2LECI+B/pLPnCopEtXzVZ4V0pJhZggyLIpVoQ+TRsMgorW1VTa2vNDlKGX3SCQ
+sLEkbGy5UpMUjJGE2A4zxvlyTb5iGseC4DXoZKSvm0uFtiZoU33l2ijsbce15aMD7AK2a+9MgkH
v+n/9c1/SELpS/tt97Bnr01owNrIkjtscCn9wfq55V2N+mcqVKpVKJSoUPZpvVGh5sUILjSItzc+
xt/VlvZoO2A0VYBw52NrWlnhkLMipJzKmHL7wKsrc1M3OVD5pai4jAVZo4aTpgkxnglaZCjE1GCF
```

```
Jwd000sihTDWT9WsVg4Sa+ElBM5ooQzOzJy8y8bCkNF/k/U1RUJ01ptV7eqoVY9g61NZs9lMRrGl
Irb2qNGEz9oi2llEUO4/l+0yjqAzNnAEIJOZaoX4AT4zwFj3mS5fPUTgOBJmE9oyEzVL5LVAwbvH
PCzLFMy2SoRVUqC9Lvs0hD0X9bfn8pm0HFmCOpSow3I5fKsbF5tL9o9aFu8Lh4IxfqCRf6nK59q0
/vte6eYYtlu0t/YxN4X8olWvbeTbIUj5HtXKRllf20/JclRarOVpcqFOdwyrbtQQg0e93BdwP5od
2L5BKs3gzHYoKSEDU3ZSZyiIjZcjCeJhqDVcYjEheqLww0z1WoqloFRuVo4wCHoemrzOVEQlOF6B
+QP3iSBWHklQD+xORs5AHUFZCqqb0Uq1pb1Z7UgMYWoTEbn6X2V9LOqr+jwJjiKfWue+sZSW6s5Y
Oy7U8iMTsKqdFu0vu1XgibSTbLkqLaoRhf4S+ehmKMt+wL60bS/eZEfZLyMzpgleoU9DfEFAFBJd
RRffK83x5vJmh2ARK1wwF43U21gUadS7F9ZUTn3Zs9w/zc0f+Np1EW4f/2Q/+k6hdhBxRPx76w/f
uGevTXIca8wLbZk/51waFv5jLV0Psyj5aJiaIyUyhWSnyQ81kqkmWIS80AxSak5i2Oi02Xt1yEdY
EVw+CK6kKEs+nw01BG7mqv4ogMUt15TrVvUulGSqhMryTYP/0cLneCCDHI7y/tuY/ShWvk/Jgkgn
rDUPzCgO7zDkcr3rFmZQqtbcpWXhmGrOQV8AWAB1I3uypPBXXS3pTGfeVSh2erzV75LqkVaMkKEW
KMlM5LzYvKYjx9bV5Y2uNxtTijS5MbWqPehQmSsEAPdXxGFjelAJ+jiNACDZqKLMjj8Xz+GuxeZA
I5HXIYyXtQN84L6AOaOay3wbXqtwX1/ED/v2n0iS+jy/g4nXf9lPxP3XBXPetPyF94eljz1j/1/R
yZqNkwqBwxUNFxNYgCUd/mqbGo46phrjBcACepEQEpWLpd6YyHG2IAQ16W9L7hN7NRmtD1XjESKw
ZHajS61U9UVN4kYBKimdVWRnYFsV0R0i8p3qtMALxLLww+SJE2HhaHQZlKWZCDeE8UsatZlj5YVk
qNFUlYvG4RhIqXiS0UwAPnOXTlgqxAW5A2wWFLFO3aBABWEq3VIliqdaOwQYk3lZyc1NFCoa9qxE
rsh2mPEdVgdVGAmz02c0WPb7Roc3BkLZFtgMfLZKCFCITzMeGw57QwA6hgM7vgeEC5PTYgGSD4Xw
339wvGrpZOFZcUIa659DIQaiNIh361abrx2kyMeMk2owBIn6KjpPf9lMiBSpyoHvG+hTa6NQosXC
mxvq52zo503b741H/d1c3t8PU8lORrEB1kw0BJGfDIJDCkQAPgJ8VbiClAndxY00NuOvwlUzVqM9
SzQII2k6ZXLHFIEw9qC4ACaEjUcwJUmAhhRZCCBiNOzK1An5fMWT0NyG9IZ5VDX5LZRbFMUefB+F
xZkjuSthsMHqGKjKfI+MQEuG0KjDxNcEIbSWEpfq/tsr9DM3kIOhFRd1Cup8rFWgtfiUFKVMx90/
DfvRhkTsrfR61GYb8/ZnVNeqHIRU5chkOQhrzJYDdcDjqCvABEYsCmhD1ex2RIUlJK+TZrqjwye6
XBJQr1cnycpRqhFUEKQ+nRJMB57eFxmACQAs/fzIe1nmDdaEm8FQeCfLvaM9Yn+oi0uftsz75sBL
Hye102p3ffeTxRx85d+HScMS7PhutsB0CKQS8MHiYXKlIWtIYx0gb/Npmq0MToKPiYJYvCv7XNsX
biQFo7VTtyXUhSg0DiNr4FX1OzJtiwQN9BXFiyX3BZSyjawqPbEnImpM8k3Ke0KhK7jmtEOOzx5m
E0/DoKZ8riyJpbcisK+kQWlOKqpZROqsiS3g8XR6CZjJm4f3uqo1V1AK1dxgmvLuttGPj2aYTU3c
8pvYAA+lsyIYie0NPOeXnYeIpkUpxrGQvM3VNqAdsb2/IQIWpI5EQkzuGGpLIFcoy+4rCHtpVMee
1guE2rB0Mpg8Gvdy5R++9Jo7jw2maOE/lsrrxO35RPtenfuv79oz1qbNVY7cV8T97mmklMKhKpZS
6lvcJDsvG3f4wG8Ng2bPAYKd4YZR2wCpoCeeQI2t1qzegbr8nJpBg8aDKC/U1LEyyrjDSTIW8WjR
ZMLzoDzqKPDuz9GKNFfs+SeHJ5NzZIy9X4a9AC9kyh0q6RyqiysAvg1kQhuLYWrdVRQYwCuivIkx
GeIjBdlMKYvreJPGTWrLi9fWYm8ptDa0Wrnq5s7aFeFhrN81Q6AGFKdYCzKLnw79b3eqKrAbuHob
e0R7DZpaCRxkjfXx9ne62tHZQdHJsJesB0vWdzYtskJb8zHXLBAoXOa+hBKSBRQZ6SuhhnQLmXZu
G54/HsWkPY2PpzOMPfRXf6Obtv/49Tyns6KZ3/oo8ns7jeULybfxPvkfSRfO8bPbVatXS0SMHh8P
R0BsHoYEFAaOJ4V3BXAggup2T0BrwQc/IC3k2hrAvb65Ssz4nFC/sqJWqG9Fs2kdEsLTnMAytlG6
p3FTgdTC8SEa3xYiNKCEHhS3xpnmyOXykYVe4nxSYwZGhgQR8wlKRVXkdmASlBQMvpwtAGCXLQAA
eDqRPm6LNgp/ZClssVWwoypkKA22YyuAMreRuGEoIS77B0LtIW8a62GSq0i/CYFSpMTKHYThDRRC
4CxvtNuekhkzP9DnsLeRrcjqkkoBlgjlDwme+Jsyr+qU5DhZKHDaPqdftsbE+QQf2H6VaY47fljc
yjmik+OUU2dDZa198SLiHhS7GzYWcLmz3ts8cD6O03mpdvinJzFft27f8Z/f9/o8GQXddNjKbz+9
XGpIe4W907Te9+1m/iu3ng6HKOsuMJ3mQ2fwEyrqGWeI/GIR3g+Zc44HJ2iTsdjrU4YdjW/q17OE
49Br1O6L0zWEzh3TxDAxwfnWDrr8qohhtmkiz1MMoU71w4WlAXD3rqWqDJWMGcsd0DbTV7MiW2Vn
0FN0cqrEcVPIiN3SIKyAJGIdImE89XSRTMmjvmHoPSvX7GFqNXQyLN5EY3tm1FUAfHgreMAYRXKY
kJqWSnan6LgpK0wpvGitgfuooQWc58WTX00rBR1WmEX6rsNag0xdWpeeazxdEpNq3xwLNFExxPBK
uKcEim6plBO8PwH+1foD65x+ikDeuxx67j05c/yKqNxb4I7B3HWN2dyy5e615lPqdNd7cKlApyGW
WawwG25wr53LslY9evHzha9xc8bFaGNxbcNy0tHhEcmr8jU78ix96zqzkL3tj3aVesmiXRM24on+
TwYVe5p+tob7juU6uXqk8cf7M6footuwk7MsfNceLe5Iqfl4QcoVBm7x8icI0oBx7wNWtDV5UmqZ
U43YhSiX5VqY2DJqhmjRtqKkkMQSVZFtiXBB9EiE4I9QtHVNADmkWKsFmTAUlIaRm+Sv7VhR6Obx
F0UiG2FHkkcJQqr2rIVy9Fn5fqEoLBKRvaWIJCgtGhqKXGIuEs47ymKQ95gzGmKkqsKY6nZG0SQa
cacBFJHzG8lqA9znM5bSZNre3pXKOw+X7iNncYW/InyegWrnKIeyIfL+gQnPkqPxyYLKHwx2h0sH
9wrjb+dNn5Hke3+8kGQlxOSIEeGQYKsbpUt5tjCSx5+YO0r0P3m3n/LnGJDRefvdddz/8optuWkv
9dN0b7mQA+id8bz75a9+pSOhiZbzl5kF64bf92LNyLT9/+qyGqS3X+Ox4GH8zNBC7V33d2/sWW0+
xmLsjTiYblmm1xmOM2QXSWnDFi3GQhwIQh8TA9lqSRzq8YFJa21xTPVUB6cczXiQshkTv5JnuaUr
PVV8LZkNlrE7CUiWMhQcOEGFDF3X7/CMUDBESJkoSQ96fQ1rOATPounJOHbDX2j71EO08cj+Ntze
kvSFAChigDJhzbpzz1dwsqTE4AWJo+KBibdSgDgD5sytbX4kyVHjnWH+ldLfwpMnchNTNzGY5eo+
vbTAAS2OVwmFbV7xNUQwARDLg+2QXazLOiJQCD2HMMAxhh0AKgAELjNRhU+m1W1Ipdgs1uX+RoKE
sgYEKWYBqz+UaCytrc/MH2OA79mA83Me/fc19d9/zkm5o+9jYEGp/HoLwZ+0Sfh6BIrLPFyRrWpa
Unvjgb8N8hq5t7xw/evQPHTOxhF0vVGRfWap7nuzdcrzjx7xQfA6NZbFz+La+vTnrN6qyaqKhgYY
YY5YpBTbV0lDwQtXGUTOhUtjRUEUB2HN+HPa2KRix5yg0aNBalUKKPIKhUHiitwjvNeFNJRmM2RM
```

12YgdCrfWhVfYwCB4vixTKShGoTBlerlZ1VWKRMAdy5ieJkWXGxPPNppZZX1qtKZCUYlhEmlARDo
TDDf01BCM6/zaKhVKRY4+fIowJB7zNWNwn6/DyxXY0Y9oPBjyZmYp6tTZKFGiSM75miCvgXx6wtF
Ep98TnuHBzppuxVmq7wnjjwSAYWLwnP260WjM01xzGclCMZ4k1w3Hyeseve/eq7bHQpUh+OvPRRC
+FwY//RWl7MmGmuq96UmeNROQg+rxZBJCsltkQ6kfPnIkl2bG3asb269ib8spnkuNalWptWGW1av
S5vqqFDuoUGTf5dNmpyv9UWkPpirPtYwpAEIYdlWOaShpSFKTsKrNI4j5VF4/NRBHeJsKusBlkld
Zlt5iNOVrYu9qZ2oiRvqvjaaihGlUyWJvZbKBCtAentBQ1y0hLobkASPUbY+pF5y2aQwz0UsDobG
lveoVt1M2IUXRqnpPWmdSNWs1gMEWBfn1rU1anm9QOO5SjCiADTSeRMJgOOb8v1ht0JCNEd65Pwx
oro6KLnvUXEk2AdzFWnWRAg7xhxw57KyeoWNXH+dz7mej7UtrDM8CybdV8tmwu14uX0wMJzdeaNY
GjmMXLd59L1xcm49SesV2f3j69KOP7CSHltYajWaC6rJseNiDeSOJg/GesT7jtiu5lqOLv5/tZUX
LQWWwadJhj5B3ff8FR44d7fg5f2t1fXUhTTNT2gquKbOujpMXjG+fvRm8HOQXe1CL63dpsdbgO5v
qPHlKwm2q/JOUjIPC0Boy35qAlIwidVWocmJcTfeILT07maZqvA2eADkr2PV5iQoiysxyZBcK0gY
Swu0sPwPdYwTPmEIMUzVrK2GqSHMoahjB6eO+8Pum043GFPVphTHW0cFui8aQzyNGK4+JgjeSYtt
A3omKMQLwIRsDqsCYtMnn8yoS4R0M/VXglfHVtHwJywdspIt2kz0vPxc0r+yNAXUyeKOMRgNpi1k
2RzqBST3OOYscyoIZESAIjCaGgSKzG/VaVtnyaqaVtYuuUTx65Ei70+qWur3RMd6AX7++tXN2cXH
+o2ykfco1RL71SoLwvTD4GY95p9VgU0+R6MWXqUoqPIXydtmFJIn/gsPKWzjkXV9cXrq32Zw7zXl
QgkJQpVighblFOrB/ma675ghVSuxTPUeqnwmHbBfXLus+oAKxT2ddSft25dmm3yeCgEKhRuFNIwH
iG5qexdDVY9sxFbMfwARsrA4vYoyUCcLYBFiC81CwF1qObqHo/JEi7QVTlWcKdFBhkw1wRDmq+gr
I4LSoNB2LE6ObjYKlGj+sIpR0OqgwY+LQI3YAVWi2Cni7Ab9nf8CbGBtCDuTh7O0RaSAvDaAZy9c
WjPszwAgA+Jxfyp8FtDyl+hy5qE7HQ855axIOG55L584+xp56wt+3NXGaIYrpqfSmzdgrVrPMBOm
qWXQ8Z9s108HJ649zqGv5ju+eHA66r+912ke2d7btKGwZiFJAEA6whiII3/Osz4LImBfHqEO33XG
3cfPLTmZorWCnN9JAOw1ZfOM0ST8SBpM72UBv4K28US4VjFbL/e7haPxK2/UcXyZbUtq3vCBh6Vo
npI3V84JPvbixTSevAjO/GqJGziUhsCUkpapNk6nI2xSYQKyqu0AZae5fUw+KUwaDjYn0hI3tKQ4
l0Ji6+aJSII9CyT1h0ArdY+sdgT9LFKuIAgaaxbMWi+qt0m74K4MG02IYPKOlI3RDSambCkYo9DM
CeDLks0iPWFe+VW6uIolEfVK6tHqJdrbWyEUeCO9OkKFMxYMpZsgcRWFCUWYL1erO9irZHJX4jqd
gl5Mxzc0v0XDU47vEaUgdGkD8c8+nLkcwrigVDGkMoDEAERLKRgZIA6Jg5KactPPfOOaw3ij4zup
NN5xY7LS7zVEwusmwzbtzfvlcqdLspfx3SyPOiDVB+J5nfRYcIvAr4r6Uu+POBwyZCIlHogiOKqy
mYsn6/X4UhFE7To1PstF+yLacRw7s3/+LpVLpz3kfn1iSY5I04ReaVar6plCGwsC2d1ZpdXtTVNx
kGsZUo2WiEpplMwoW8V2KDFBVRsH+QNZMvhGhqjklRLMUy4RA9oWH11eCwwBBoH+Ldk4QKpZBy9Q
V21A9TMU+KFM8pGdNs0wXsqyZl5xtaTICNmWfiCSczQydpxqahlWKaKkUzXZbZPDIji5C2TKu8Pj
5c1IQArtkoTYvSuY4xuO2MP7Ds7b6Ia2ubdI25/u2V5b80ZDwOJR7mMv71FjcR7YZ0vLCCuV9DKZ
POOQdCx46EW5gQ4pjqNoP+j1LKNl5R+H74rCH9tiAy5blliuVemdpeXF89KprcnOLS4fjOJpvd/s
W0gd48XyhyB852POsz3SzFUyFmO741KcfwsrMvfylJzEeR61WW/Im6LpEUSxTMMhP4Qj5jxzyvxu
ZgsC6S/NzD4eTSYk391dHk0lulMVsGxbVqzl+TY1cDlG7HPJdvHyeDi6t6K1QVYETkHqZu3OfmV7
00++VcScKlyucwZaO5i2Vq1KmoYCkNxVbFmicgYyMxItaNiB3OcX1pL0dikuGRjKpwYSp91QEa7v
tmWRGvCb1XfBRAemEn0+nwQCA0KkDDBhGYcgo3G7qL3V19sDBJKIzZ86pMTa3QJ2tC0J0FoyGVKu
tiIBAB2Tq4E7OoC2EIhEbMPDCbLBo9fTbq1SpVYXzuMSGlDdzNBh2yags8mcEmCIUEvZ4kkp0Ijc
8A9IsNqQ84bhZNh6acZhBqsQ3bddqr10sXbiwbnOU8oLl5QMr+cRYG3e3xiv7DsZTgvA9z/pMJrB
QAcdIlcodgUgYTb2BjJ3xooTRYi4VVC0w1igMs3ASNiaTsMI79sJkEsxxrjXmhfX742H3/xn2Ox2
M0YWRMry8Z1Cx4JGfK9GFtYvUC0ZCGDadppkWiaZfU835KyFjSnqIPJL+KYkCQKbxu1N2B1e3izl
vzdSMK0jWUs7dkiHneRwKpgLcT3UPVFeYMY8KD6q5kIS3MFFgfhSf5Kupyc0wApjFMxoapbGY0sw
azSmz4pPH/KazrZKno4jGz1/bWKfNrR2q1hYlPwWXMNICFIRwBJybIsqplktULBWE7BteFe2biQw
cpJSvLtI4GMim0Vg5RpVKgVZWVvhcqfAtscXJe0GJHr3YQBTpc4KAEi4nTCGnmXCSR5MQf/dcbeF
Aki8U7Hy+OGeb5jz/+d1xGJnJoG1ogvA9z/pMZqrBaKBrsLsR8Wy30sWNKU0oG4EZx5BvzYocnpb
4+6p60Db/9g5e1acchxNXMz3B2/m3hVHgjQZDA4um3qgKq3+vH9OFC+eoeuIkOyp9fgk9UezRGFp
DkXvDLjAGl8qg9RQ4obRxYCI2hrMNVa1VPMUZ53i8wUD5fAqsYM9lujmZDSXMiNqKLtQwdHtIHHR
eFnCa4qO7grFVvatUVXYz6fIqAxd+KOCWU8EKywVkOmYXEJglM66ymVCiYcOWOGB8DrzD3ffdL1B
AGCB4j0F61t5aJdNvCAAC00M1s6rYIKwiRwgTgR122y1yGz4bs0nlWkVC24Bz1jx7vWLjABvuFm+
oHbp86QJ1W77Q8CDMJnamwEhjCMLPVxVKy8xUqJxEcgc5ZDby+Tx/5MC0rVze852CmSYub3DWmdX
t5OD+/YkLeco9Y30mol9DdvApWPuOux7AHy168lOkqJRNjSROEgiKF9kwFnh5ljNYF9EZflzm5+z
AKL/5p/7H6C9+4pu/2/L8ZWeSvCKJogoWth1P2GBrMkN6aWONrjl6NRUKJV0dTVWZSNQZwesVS86
LXBmMiQILFLtQWGRp8qTQfFUMigY8UoAxtwk5hXUOKRsizwjisYx/Hq5uUDi5KL8Xgm0QvNUbos1
q5jwF6jcUtDDNxnoiRlWbI4SqooSXCv+xqUPlJ/HjyqaSkcL4G7tgiUyx+EstKlGsEJ3hSIwV14Y
BBY/fHxVg/G7U2yCTjR7Keb7vcrjbFLig4ZSptXGRQ+EJlfILHI4WqN/bJL9Yo+E4Ir9iUzTYIc+
v0JFjJ1W4HisaHHjZOAvltaZQ00zkvmLjFHoacEBNhvw8n//SmeF5NuVcw8q5bs71PGs84vw2aBl
pWDe6/QvZnrE+A4bKoY9SZ9Pom5tuvD771KcfTFUoKugigxeQdE6kisnxL9sqG2p2gL9vGNL7oMv
8WEU3QRu2rOA3v/f98d//4jvfbNvRbY7jvCxG/qhnMhvNBrV2WrTV2qIc51qmgN/VEHyiih/K27q
eYvyX8DGTIlMMVE1iikeOxwGZIb+OvWXCxmrFqmAV9lo0XG/LqBmGucEmgTxRsfHDk7Mxdnlz2Nm

```
SAXUlp2Eo8IVrU+Kwn8M0DxsLFjome2SYHkCLTBuipXAi8JxCb3qFoU7vr7ZVyROlK6tbVbfc8Qk
KY/bMkx5Z1ZrM5eL+otpaZyMtcuiL90I118gCqtTr1GtdpoVmhRbml6hczLORNgAVpDEbvm15Ir2
R9/NS1cZGMr94gNobF+TzeRjCdzOpTKN9I1EUtHM59AYM0cLUEiKPGD3dspzHxRgQaudGnM7XS8a
wH80IwveM9elMxEXASS2oRAimjVkh54aTV2WqjyceK4WxarZ4/Ae0+YJpGEdVzEqP8zkuWqYZqXO
k9A0//v7Zzvvaf/Nb8Qf/wzu/wvXy9zgZvQDvF2OQnI2qUqnQ5c11Wlo5QJ7pSENDMkZDgHnC6YQ
dHyRsGRTNcX5eaKN2l3YubtDahbNUK9apXKiK2jg4IKAGgLYJQBP4XMgtB/aqKKslwAvH0I0pEWq
xXqXOnq2stG84vOaETnRuYinqBDQOOdzkTSW/uI+vr8jPmw7Ik+4Rq01FcQ7rPHhagDLSGZULvk/
RtwVkkY22zXn/gw88wBtXoIbRER2gIMaGkC/4nNPnKefZki97HKaif5pEQ7rqyGGan18W0eTJJOF
Uoq3mh2NsPNACSii3dEiKXBkYHCFahSEA4LahMwueKXH/icwZB2zovl8mI1eQdh0IxfE5cI9570o
8t5Akk3FsFaqpW6yknNfwRm3w+Tp7xvp0Ht1OS8k1xPFu9TWbAnCm7HxXeGEs0oTXjkFzvCavgsH
ygr2fv54H7S2mPFLdXvns4zX/128lf/3Tb39xEAzvdDzvxXmnIrlXkRfp1uYWtdrbtDi/T6F+NBB
DATOmNKF8bezp7FEoMo8oQxerTVrktZ6vzEnYF29e4vA1IYsNw7UyYUmwDTWHakMdYMJhn1fgsLj
Az/cVpy4bqzu3rHijABaAZgyp9o1Ra1BSKFDYvkzJpK+YKcxolrfL8DgKU+YUb6VYH0gzXKhZV30
fLYWCQhsG9+e2j/81hYHKU0v1fTQa7rABJFJtNwVOOSCX71FzYZk9LW8q7NUrzcNUrc/RcNCjcRs
shyF1ux3J2UvlulScwdTR2lmlenM/TdCXlhlCzlGzkSxiKM7l/KIQhSecCiD3nIiItCnQRZS085U
F3gwv86ZRQy084Hs9tBw3cXKSx2Zxb5BZtrtnrE/nMRwM6R8OnGdXdnKepC0kYEDTrGWZeYR/wx6
STrOxPpaw+0KoBu+LvBZV47/+mX9Fb3jPf33Smd/w7343+fOfeOuNwbj3p26ce7Pr5UxoiKJV8/i
ZJ6heb/Ki9AjJsJHtQh8F/ICZWAd8Too0DFGB5+SpcfAoh6dzNN7aYENiTwwtGDY29BkB28uGfXL
9ApkFDu843Iu3L7GHKZBdrJDNXt1sLpJdW6D4ifugPkCcqHE4WCCjVKC0kFd6rdYiG2tBGASnE0G
JsDzE4sGlqAVQRqqkM+Ta4VFRqBKvmgpiMUvURnb6zCP0xBNPCPughzxy3BLuJ3g95NLNWoXm2Ei
LeU9sPs/hcLm5T3qb7a1N2tle5/cHGV1HZlsdYYtgYyzVqN3uSP5pc2oh/EtCajdR6u+OJeARIMw
cL59x1GJMghE/ryhDA5DtsHOlbDLu80eMTJt3iTAIdyyrtBWzd7XtRpaBnYLTBxCE7xnr056zfk6
Fm90vs7lWFJcsXUzJPDXbSnfx4uvncnnlVQTgrw3zswx1enzDe/979qvffvNb5haqv8HO+Dtsjm/
z7L2gpH5p9QIdOXhMwQM1cTZG3UgDhSyINoP825rIVI/jIegdsTfskFercxjXFfxsHPRUlZcNL1e
bF4+Z9DbJqJfFs4oGS7/HMZ5DRr9L0fYar78BmWzAKYD7RTbsakExKYp0JG8UNoR+EKKrdo4tcCs
F5ECPVQbdrxDBUqp3maoAq3smLaf+aEB3c64Ko2+wUSZ8HuCCi3xtjVqTFpYWqZRH9zihYrEoEps
Ql+7trNM259ZQns+AEmGDAc+VoLLQzuGIAIyHGEPs98YCxCjyvYqQx2vFPxESwygdRLGhf8N/fM8
rGJM05LTEgapgZoch+HL47+zGHOwO/ULxLEcgGxSHEUc6KdpDU4LwPWN9Rgz1ymLIrprck77XnjZ
Ns03+oyH0nfDi7DiOmxpaeEYd6YyA+/Md7/rPn8x++R03f8/S8vyaOez9WNGumVhwp86coqWFA5y
v2WpTkNIpqVAdmZlUURNF7SIUppYwLhhZj/PZNtunSelgJNXjoMOeJeOFzqGlbURkNw/QBLOeoyF
7Wa1zOhxRwoZiLy6zYZYp4fe1SnnO39h7QCJRCIZDATAAAG/4saCgpDdLut0jfEu4TlvDqLUE5HT
XSlONRkyFO+nBuz5BA/b28Jg5zhNtK6X9izWq1RdkOF9U1vmXeZ9zcI4Meu1VNiSbtkCKNjGEhA6
5OCKZQqlEo1FfvKXNoe3m1iqHsgUa7/Q4rG0qKUnMu05Q1BurMUf2tn6pzuFxpnvbZuabvoD8R6G
TDfodzh8chOfsSc0tst1TruNvm2Y0icddAKEzIQgv79sz1qe1wGQ5T1I732WImDrXJ3vdKbEDP38
9y7LP0apVrZ3X/tvf+Uff+/t/55PJb3znK3/adtwemfYvjIPQhB7OPffeRjff9DVSzRTdc6Ezgid
jAxAwfaJyRkcBHsxMqXtD6dGucTjq84IfsnkFoeCJ47CjWFo2z8vFO+yBMaFjAo4Hi+HcNTMDysp
FqYaari3AfeUVDen3GloAy/R8AUXEwi0caQ4nxQmcaSoXVTDXFWA8B51T3jxQAV9dPUutrXUqCoN
GRziW9++/WmCRCHHjsEcebySo8IIRMmxv0JgzgU57nSbg7w5j0REqeKCx4Qe/vtCYo9G4L/WsMhu
vbXo0gUhWash8MQAcaM1gsCEIJxK+4/Z5ji/EMi6KAbad8TWljuuY41EAbBXvufHYMO1ThVLtft6
L+qadS4NBN/NKjWxKEL5nrE9rNfizx+Cyf4yPVEu4fNZryPiSem7f/b5b4//0PV/9Hx996P6ak8v
9SJpYztrWNp099ygdPnxMeJuUjkwqvSFDqtGZnmLJZtoyJsbmIFRlW6JdahQ4fE0KZMmstxpbS1G
k0uz0Arh3ldJ4xsaKcNkUPl5TIY2EwV+Rk0nBC+0ZiSMxtYNcNSQFrdRarNObJlIY5qwqJ2BGtJn
YUDutDWqdfZg9ZlHGyxb37SeH/z0ebFDY3SIvV6VSc14QWONBX4Ac/cFEvDDCZIO9HwptyHs9Nye
FI/R80YPNFRZoPBxQsVynHuex6XBCIUjPefOAYDYG2NNI6eKAdQKbgzD88+snjmVYqZH5xZKAs9T
EchoZ0WTT8bw7eUM7xTlwEI87KQSwzFx5RhD+rF3XX54Z6xdvY7sh8T9krEQr4HU//Ltf1Pm+4z9
+dDK/sPQzvI7+YPPSw5Fh5ujcuXPUGwwo0WGl4IEF25vNSNmmco1iXFrdTSQZzUTYU5y8Q041T16
jxh63QnaRF2genpQ9ZyFHFhu06Sl5jV2uYlNhgDXtKOm2kUwEyd2KNVuGIi83Zvqjmv3B1BX1qfB
VrEjIUb0drj4hQwVoL+WqdYkcoI+KcbjmvuuoPLcP2ScNgURiQ9vaOE/t1gU2Qs7DDYfPo/RqOYc
UrLKLcJfzcKCRiujJFssiGRKzQYF+ZczhPs4DJYFwoobGE7SPMgXKiMIQAw4IhYT6BRSVSZqao24
nC4bjnuW693nl+duyJN4yE/bN8TgznVKmCcIlbN4z1ueCiV/hWac4Xix45Fi3/uYPfNHne/uvfDC
Ym5v/wUNHr729u3U6GfGCfeTBu2k0SYT4SeFzSWvVqGpsNpWNnF6LCESZuwLNltJwnc612mwoefa
efh76MDnN1O+KISivGKvQFjBDzUBIQkyWzeps8GzTudup5k02ZYRM0E/dVVtP9OQQBr1jzjsNr0h
esUo5fgxa29Tt7ojHxchafzimdm9IF1c36NyZs3SZo4txyFuQU1N6NuyJS8U639+a6N1g0AIt3UJ
tCUgygRGOo4TP2RbxKxgseqqZSFdkwt0cTgK+llDybBTNwPYAMAx4lgFMG/Y7VjyJzGp9MSiUyk9
YpcUPc+TySHX/NUGURnzj7ewKgnACQfiesT7HjNbnPMkrFNiT5bUxZXTLb3z/F3+D06BTrTbeubD
v2M7W+nkKOOS8685PsMFGSr/0CoCB4IZFqNmZFbOMqRZqpom79XNBx4lw1gIKyMlJYcmBsQqxk6N
G5TSAQdgTxTLVYLhsRsaTk3bl1RXo30in4lacT7JHB9AjBmRPG2oEkeL+tryP4HExAQMxav6vvHC
```

```
Yw/CKMOr3OE9sdXu002lTqzdgj5sJVzEZEaFSbkPYOQtF0ycYD4SdcAJML98FqB9sba1Rt90lDNF
gaAKtrSIYMVB84+eiAgwjQ67d6WxJDp3zy2yoeXh3Y9jaMPH7eqMZNFYOXHJLcx/N4uD22M534vb
F9HMQhPPm4+8Z63PFSIH99fIFnTd+FnriSzje9ksfyZKwfbZWrf9Rc24xbm+e5hzSpwc+czuNAca
XGdZkJpUoROCaVFvEpkyt5Wpoik9DK83BEwtPcCSFJSERhzq7o2ZO1aSOrfrJV1SxFcWMYi40dDE
FxrmraDfNC5S2bKbhh4meFEowABD0JQw1tASIYZd4E2IDrC1QfeUE9VqXKACOmWNUGM7yvqPU4Lx
1zK8L2TD8Auex7ImrjTpVKzUJoXGZBQ6BPc8RoH6vPxKh6gHnrJDYxJW5/Lt8zqX5uQXK511R+XN
0bxrE6fCoAq8kRSCH+9KYW4pKtfnVQqX2Cf6gH+qPovMb5x+PFUG4S+2NU6qfLLIihlI32DPWZ7+
hIjd1BAH0uetuX/ndv/wlnbvW3M+p2XCzXJ3LsLgs15RRvFOPPkATUp4wUYIu4lmV7OOVVUmtuQq
NVVRndRFFsUukqncLI9f0p8pYU6XjCg5zGZHb/ZMnujQu50yzma6NDMNrzqh41qGJhZ9YQPrRSKn
aSftHLWwYbSDiUgnN7b9GcMYLh17I3nKb8808G6BFBc6zF+bn6cC+FZrjXDsKe3yensz/5iyDCp5
BKytLVODn5zjPTPhix+OAPR5U01NBWLkyaJ6IfEg+n6Pmwj5+fUKVEufuSBNE9Q4RvEvD7hZxPpo
WCvmwUCysufnix51C/c/4nA+yoY6RDYTD7RlBOFQAQCWbacTbnrE+OyzyCvjhDH8oORua4qgmYqT
qc9Wf6B9Cor6g4473/SB1uh2q1eY8jolNz4H6WUzV+UPU3Vmny+eeoDBT09zGFWHqVFTrypAVA9Y
ZKaHhKTWoCVoYyEPCoKdq5YKMcjR7g2JSlN6k5KxTylDNnC+jb0q8CrhaGSBIDfmKRyY9Vw6jOVS
FRg6p4H2mGxsGA/ZoA1o+8VLKlascjpucP+dp/5EbqFywqVLIk5sEZCdD8j0O1dmw6rUV8dz97VM
COVxZXqGy77DHLXEYHAiMEDl5JNqskeTjwDgj7B1wnowUIWdhE8hTpcI5s8v/Zm/tuJ6Q1zm2m/E
1BLlc7oKbK/ydnSu+f9jeuvfcQ5/qFQqVZGm5nkGqM0Lo/SSC8GdGd/V53bp5zf/9n5/58rI+Tj/
2qGlb1tIkToqVQrHTCoaNjBeh7c6rIhZvDm5tiaLuDi/cCWnokGJDlBQNivxMkZTHbJgiXcyLGAR
fBMJuDkGtxJpJcohtC6TR0lQvjrSHppqrQD4Judi08izeNhUDn0lOCjdaIgz+hozdGRIBwHBj6b2
qIfaYF3nz+MvIbR6kkMNXG4DBUpUyTiUmwy4NO5dkHE4KUJzHjgdtGuycYsPNU6M8T8VKTYYabHA
IZxw5jAJpDnngOmYPHuvidQJQBV+OX5mXgXTPL1B14YAQpTU5xO71h8I6mUZBWihWB36hyPlG9lH
Dzf8Nh94PbF1+olMsN5NGfpytb3TIYcP8un/7vufUurZp7/gnHb/67TcDuTT7/vd+4HXClgeeJtM
wXMd2XsTmcFM8GW8U8qVxqdQUWYiw35JV6JUqlGcvmzQPUefxT5IHPhnQugjUz1KjbzJrqhgvUEB
StElq8FzC4ikAHx4QEz78WoR4U6Zk8aCkhmUAvpBRN4TKgu3V8heAQKLgA+EpeBcIbkFxINHeV0S
UVThucr6XssGgGJRbPET5xRM0AfqIc8+QnwNCN1SEcYUF9Dx76+LF4nEoBTP0M/N8fcVSkT8ne12
nzF5tItuMi49qpZLzom6QggEjA/dUQoA+5K2mTNqMh33y+fe50hyZbiDAjmG3lbj5cpfv/X382o+
k5H6stXb+DH++XjzYiXP5Uma6E7JjVU1/rh17xvolHr/2Ha9gW7IyP5czfvf7XpOx9wTQ3/Rz3iH
TtBqmaXyGw80JG9Y856n7eHEUHdMcoZI6JkXwhcGAzto5MtwCtZ64lzqtTfZCFcHOSidfE5dhgU9
rXTL5I2LHqYymiYLPlA0RhGaZgi9mGMBGg1JXdxV88Qr2c8xuo4iGdqtpCpZXhuARhgIkEGseJ3N
K8h0rXWjN0igCxnydxYM3iBqciRE9pBKZTxMTraJMzpsaiuwtV1ngc3WElT+fQ1W3Qjbnno6pQB8
2rh19X/bIkr7ztWBGmLc7QSgV/IKIhI1HHaUxxDlmDhsTwuRSg1w/SdMwaNmefzfvcH8Txsmdw53
Vi5zW9NPJKC6Wylk/7AsBW9DnNKR5YM9Yny9HoVAAdaVl21aeDTWxxZtaZc6nvp5z4DGvwUfYoga
84E+zN3o0i6O3OAXfLTgOoclByUhyMJBvb557hNZP3S1h4Lmz91KZw8irT1xLOUyTYIibQ01pp6S
Z8qam4k8SJXZACA3lQ4U0LAP+OFMF4EQhoPB76NsolJPaKIB8gjFJ1pugf8rvwfmeOdGM9LoKreQ
cDcmHxXmDWSIcCbFZ5dirycT4IDnkRCOZqQVlSzroCP7CinjT8DP+bVURo3mgYalJaJ2ELcEde+W
mfKaYowJo8YAWxiqUtLCXIVM3fs6V7/P82ow/HHrNjtDYOJlfqhu8GWS9nbWe7Tr3WYbxV3yeO6P
JeJ13rGE47IVRMEh2Wp0MDIY2581uYtCL3/buPWN9vhzv+OW/F4KWD/z4Wwccmqamo jHNceibY4+
6wY85/v2N/HiQF00ly9Kh49qWUa5SORyLUPC+F76Kzt/9Edo69WkxstbWKgVsnI/ffSeHkhYdv+Z
aNYEzgbHEiv1+qjmjCc4weSIUMJYKf1GwgaSEcCvp50oLB7qswmOcKWEooIfQloljIRizwkAxTZj
WFQRvuqBkuZLLilgUe90Je9XyNV9NdrkuBRoT0y+OK5Q0JlupxZGCC2ecIBw3ZWA85fxz1BpQa2e
N6nNNzj2XObJQGwJ0ZgxMFMGLs8fdWb+Aa0rSycTw2IBz1eWk2Fgiyy9FKX/IeNxLeUOMOVqJQW8
a9trpsLV5ju/P33nF/F38nA3+XX8y6Af8mZJyuZoZk4wmHIpb9WVyzeA5ueb2jPWfeLzpx/97+qG
fe8cU/VPkBwYCunJvDeMhXvZtNpj95bL3rsEw+TM373+f43ZUHgkdUg7N4JEgJwmmxPGYw0uvSFt
bW3Rgkb1Pqc4hXpm90UiG01MJSUkXmlxFPijUocJbqNsxnNumWoEc4HRSbY00VbKNMusjPVoOdcE
omChhrSzdpcAR52pau7U1/t2YFztY8es3voncckOTi+fEQzuJjBeItGPmJbIB2K4rda6EvWDOc8g
rDmjInrLV7cgmsW/ffo4c0EcZ06i3TZWlw+TGUVyozMW+4fQ5lO45hWKUKzUgMz3meGKITY/fdjj
pbo8DMHm7Lli9+TTRJS+Xvy1Nk9UsMwaAO1kcG2fJJAuNIqcoE4rRH+ZQvXrwuj1jfb4epqWkINk
kriGRj8QwqnHKEHp8qvIi+s5JYt7GEZwD6OBo2KJSZR+1L5/mRT1SzIC8uGEoYRgJeL3LoSSQSRF
/paRINudshkyaBYrU2xSiFyXpmIDhXrViDC0tiT4pwmEC0J0UUAL5HSq9ZqYEnxQyihTbfqpy4Uy
/XgEqrFnbCAWx7s4Zarz8W8kuNhTFi63CU7wPvPAUBmk5HkfnnlCPptCGJYTsPhXBVoh8mD31uVO
fofEklumiY1dfTfninHjiUn0+iOP4Qntr+x7TdB5PI2ene/GhESpDnBtHSTwJMa/ICcgkisYTKtY
ndmYkrpvrQ6IzHLaHHBdHXr6aGOkktf2FFCirQWudCtU5+bzHX/eOPWN9vh23/Op30Ve+6zcVpWi
agg9kkRfsg7xq7zGUqvoiL5R3JUkUh4PRb2dp+uo856mgOjlw8tW09sS9wiMk3MUuqEMHnPO1JBx
EUUXCTzY2UZbDYDUbg5Mrc+6nRKSiWBGUsTNSBOAa2ztlJoQRgWM3m6hZUglzpbc6bR6bWm8qVhB
F+anq9aICHUt1mDeP9jZ1t05T8+VvIa86Twau1bQ1JFLxL5muqdgSgU22YvG4qCinXkHEnVEiy0U
NqvP77Fx4nPZf9WK6/daPUXNhhYzL25SNO3TVVSfi0vyBbm3u4K1G5rw/TuPHJ6NO6Nq2kbB9Go5
t8fWY0XhgxNhPLJsj4p3E8csxb0QBb2RxrlDhy04TM1cU3lYMEmBMb9xeVzWC5zC0YM9Y/wmHGCp
```

KMHEI0rWEd/w1NoG2oeBDL+f19PpoEtzJK+RTbBIv8fKlw/3tS4Jz9SocRrKXqTSXlbGmHEpCSNn
wBX20s7PDax3D2HlS2rBshAiZIw4xXZu9V04EkoU61DRV+8ZQquGmxhaL8U5bPYZiUzLT3SEFGck
TfO9UhHgirVecUzC3/Bh12wLJm3/5vyB/8ShZecVYL+fLtBgVWBPZlYPFAf9GFRqap6DXlmq1o3q
2js8hN29AcXNA5y6e47w5T/1BSMXAoE/f+RBdbEXx15eLW/sOXX9Xrdj4zKS31RuvBWyoscmhhTH
sdy0zXzKTZGIYpmewgabxZJjG8U7iF6tpqbmUublcwr9LMSOcqy1yDm9S69LjEuYDPQYe4z1jfR4
fnOvx/xMrTZIz4BHkH30tL+VjbAnv40UN+bYF/tnBvO/fOB70ZBh8k70q2h/VwlHe8XNksetx2hz
y2gp9BCB8v9eiufllDn0jJViMCigMMgIZ3EAxS6A9AughNHCsnPQoZZYOXR+HNMrJEMHnDNIg8KS
gMEnVGF6WTtFckaZKJSlaJWHI788edfMCzd3wtVQ8cC050PMxNdk4zmtobDEpqtUpZHKqLGdAaZ2
fn6RsKMGYco4r86jgO+52+2LQnKZTHtIlqUUPPnCeVhaao6Xl41tueXHUefxTiVeoYXgdu5BRKFW
i7fULqIpR0O8Y2GiK84eyUn0hW7j6pZxNWFky6mZoNyGHH7cvci7co/bqOel7O9Um/+zynrE+n4/
XvueP6a9//J8laRq32WBzvP4e4B//Ha/sBi/cBTbaY4Vy/WXxJKhE4ZC8POduoza5nk++75NTniO
HF1eh2yXAEaNJSyZw2JNQc36/RjPQbhhraS+JdoqMzk2ndFSLJpO5VYAbLCXbiNQWgHet/mbFqoW
jRKgiAdyrTSeS2VDIUYC0bNjZpMrxm6h8/EZywUlkKgijqDvqHFi8tJaCNKW7pKCICIUNSwEPTCo
KyD4Hr8znh7LcVmdEo9FIKEcXV5Zp36Gj1O6NacjBRKu15jUbB80APSUUn7rbWXX5CHhlqVCZz1L
2qgUOvV0ASmpL4FXKEn4ejJ/DYhq31ijobstsLCrXUEbPl5QsBiKGPWP9Mj1WP/TrhoSHpk9LX/f
Ozws7ZGNBoArGLrARC0d9Rkadl/Y+1y98Je/5L+Ycs4TQUiQdeHE1lw7LLKjNnoaCkTAilCpbZK6
tiiFibCzkXE6mUrKpUpuaqMk0telUWEoKQvC0GBQwzNlsqtKj0bKMgt1jT4fvoXeTaIEqkfFIBDA
fYT503KMxbxSlw9dT84avIp8XOhgKDeEBTmgGk8Z7pkpXlrSXFWK1OJEc2ZI+sCV9XSdXpMm4z7m
5I2p0j546S9fe8BIKLxKtbe6wFx9Qr9W2W61C/fy5i8cvXN6u3fCG7w7O/v2vpSg+cawrfVLk562
trYxDXfa4Y9p89A4JcZEvWxLquvy5MAA/5Mi5xAZbpXx+IrjjoL9JX/Gu/7RnrF9ux8aH/z9zMm6
Zhf3XZbml67MZhcPnOV7/Yx+Q33/gPa9RbIhp1uR/XJ0rVb7B93IvTJK00u/36zCqUnWFXI6OK80
VDi+vpygIBa0D9v5CoSDMCqDyNMTLjYSD19AiyKJKLl9V3xOsE45AEWOpDqdRptTqALQ3FAIKHMD
yb4AhEpWzQZoDbPUYgheBKzY6LHBQeI45BC/ML1HzRa8lrzrHUbm3q3dDioFCpOtI0cqkHOaCoCa
bylySUnSToDox1Lgh4H2mmkvd6bQ4YligztYWDQcBtfvnhR94FJP92Ln1+srC3M2FSuW+n3/393z
sR3/t9wfVan1275eez12HPbN88tG556+MzVt+z41GnWJx3/VO9cTXkVdbzvJzB/9RMP8Hf+Ytlu2
4gCFavKBfzGHf1+a83JHMcoN269KyosI0ObeMafnEjVQ5/EIZ6XJyOfKg/cJhHUbOLm+CZV8ZJ8J
eFJowWWNopTYBLYBBwnSF7HvaF82m2jSpogeVOhKYKOKAPflESM9BIYOcVBgVwKkbBILDDQZ9moz
4Kz+cYoWaN75R6E5RyEK/1TSsKyhvMlWFBveRJoaRa5syvRqKExk9XnAbA05oOnmZbrHcPD18/32
ixN5oVmlhgb1lEBNvZOwF89Tr9YoPn738wq12701uoXLyl977ntzeqtzzrJ/zSAcdy87X5uLeZi3
q7mx2H7u9axRKEY1Dql73lTODPffR3zNouG0EiZFhkgR9SNfLgyZwHzuca8vF6lvZWo5lljfeWT3
zCohFAerWKC/QsZe/kexiTYHuEwVe8HyMzuXpoUceY48HsUJHIHUp2PLZC6c2DEDTlwAza6sWS0b
agAXDq9TgIg5J4U2l9ymyHyga4bmReFQwKoAkGxXamHNlgDJQJUXPF9XexZvfRIWF/WRxTi158Yw
3LrvCYPXXRCmgpzE2gJHksdDugaGi56qEhQyFvuINZnV1nWqNOoWpSQ/ddy8dvvoE+dW6fM6V/ft
o4zLZ6+vrjVK1/Kq11a32cHxx9NPv+ZFH/t3P/Ptwz1j3jicdiZnaWX/zSH7pxMnx6v2XHC9/2rT
sllNdHm3f+T+SzM3x2owtGq77RnE5c5KYHdbET6OwZHvewXIS3hhPxq+IErORpYa5sXrqFaAoWd5
/FTWWDtHSiZeTC35bNkQYXTyBuvhYPJhlj+ns2XNUsByqVEpKc2am5apY723LevIsrqXlBQyF90W
RyBQK0UjA+1mi9FYT0V01BHSfSplJvTfU3WCsMRurxbnz0ivfTIXlw8pQVSVJGyYKSaaIP8WpwgU
DNzycDquDdynKdFEJm4dq10ixC31fCMzwddzxqVvp/gfOUr6coxfddDOH+QH12y0lBXv+AtVqNb4
uj2+yfSBOjK/v9frx3bff+sfvevv/9uj3/psfCK5+wcuyPWPdO9QNqSxbw3N3lw3Le2nm1b56tPH
IRWPr9GZmmZ1kPB7Z9f3JaOOUb1dXCkY28bPJuGnG4VE7ncyn8bjqeIUR+5VKMhoeGAwHHhbj8qF
r6cC1L6PqynFyK/OC6snCUKhS5D0BfnBNXvBbVPBdJZFR8MXrZtmuELOpVegsQ8+3ZpqqVED2lib
ezqSZCl4iwBNJeJOm+WoiI3DITxNQusR8DZESc0ZOOf/S11C+uV8KQ8bMUBXCSbSpDKViB5JUkJu
lwpA4HatTo3rwoMBJC80MNg2JwxX16c52i+6570GCrM+5i5doa32LvuWt30gPP36Gw+AJ9dodSng
DQT90MApyw2H/0Hg8/udJZnjp2TN/8ee//9v3ffVr1/ov+7o3p3vGuneAlc/KH3xJPDz3aT8Lei/
OLOumJB4G6bgPGNwk3j4H+J7L7sAJg1EuisKCW2i4WZa6nH9ytupm496ON+KwGIwLh1/wNbRw9CQ
Va3PkNfdJrpYihMW85hScICrmE9FirRR9qjbmKRx22DjnFNUK/94SIWXQmijVNlOGyxULBAwkIcV
6aAOYkMYzMS4l/2FQBGhhpMbnEkE+ZZILq3NkVDv5SiosHiQ757NX9ZRXh/xkpuZe1Z6h2kMzoS8
YPopUuh0iU7G2GgYQwSqwVGhqTyiZf/wjf0OD0OQcOaDa/DI9+ugj9MlP30+Hjx4S/uBL5y9JfxS
5dMgWze9TNCzrmMcJPHvsxuOnzv1lp/V7n7zr9ls2v+fHfiHaM9bn+THkhTy+/HjVzcw0SiY5Dh3
RoLNS00kyEDO4eQ4PcxQGQ0tQPuwlJ9EFyy8to98nWqRWeZHm999A+cYh8gsosuTJr8wp7C5oPiN
DqDclTBQCQrXu4ElLlTL1ti9RYY4Nlg07r4nMhGM4VWLFoucq9CyZHihXeWwmjPWRojON412+X9G
5UBQvCYwUrwPnEED9/PzqsRdR7dhJcgtVxZYoo3QWv1wZp/RkMz3ELiF5osNhzQ1lqCKTDMnbirF
CzoGCmu3I7zGocNutt9L6xjY/RSGnmgvztLq+IwiuCacKCyv75T7sbK1LAatSq/IGZuQm4+EBzoU
Lk3xucRyZh9fWtz/8Q9/+zU/kipXBT/3K+9I9Y32eHvuuf2Xv4UuPrmRucb9tgDLItCb9TYsc3zI
TXyhAY3soizpXnuevKreEYpnBeahTnKdcfUVYC410TA7nfoXGPlm4MMY4UYUWtFmMSAs8papIBJT
RzuZFKuVrYpSxNmYYlJXYMuKWCjOhwuEqlTc1qC2OT7opKAgFangdfVSYdaLaKKISl6oh9NhUPN/
FA8epev3L+LM0RL9VEEemYnxBzjs12ClvVapJ1oBLNrR6HNxuqnvAln6eMnglwgxitQsP30/3PXS

```
O6nOLgo6aBCEVSxVau3yBDh65mnrDCY3PrYkOK/rKIEzLgVoVbBB85MvlhVKpVBgG4cpoFB4PJpM
Puv3g7v/9m9681mw2g//3N3873TPW5/Bxx60fdzfW125u72xcvbF6+aQRDQ+P+v3FlYVmvUCjw4e
ufQE9eM+dyaC7nRw9fGRSn9vX2964NMyGm3T1C15SdvxyEXxDAflkskcFltepz0mPEbmbJ5bEZlF
ucp5VlH6kV11Qbx5PZNEV68tiqMgJoduaomqreXlBhSkTNJnqh4o6t7D0J5KLokIr8ArO+WAU4Fn
K7Ex6qjA0y7Z2q9jaQITtUAwnU8Yk54u0EtzUSysqF68+L6CHPH+1XV8po2v2fkl91ZmVWruM5il
wRZwoD4+815Br1YPxhlJMn07tpBEkLnwZ57uVvSpy1f6FVWrWSlSvlWXYAKgmTBqBrA5s+OgxT/h
6JzE4lSbUmKvT6uXLlHRTe2trCzCqXLVarXW73aOmad5p2/Yto9Howf/zW79xc/+hA+G7f/aX0j1
jfY4cP/+TP/qyc2dOv5Hzy1fbjvlK27CtixcvCiLnqmOHpDKZdx2oHdGZRx+mhYUla/nAUVBzcc5
p+r6XS5aP3sxOwaSd7TVr7fJ5Cx509eIpgfJhJrJcrVAQxXThiYfpDd/yXVQuL6nBb7+kmRhiUR/
32XDFowlWN9H5YboLrM/UgPdU5gKyEKNBn/x8VRFsJwpVBP1WdShvKcrp4GdCzmspgMM095WqsHi
/KeWoghKaiWLTV2AFzm3zBWq+7HVUaixK5Zc0Hako2SA8BldTpnq2FmmZDan6KsJvcCZJ4Up7UkA
eETGAxM3UvV9RqOPvgXu+4+O3y3VN+N7s9EfUZ8951aFl8ngT7LS3pNUEZNYkjGVjCr2EBoMt/r1
Lhw4fo8sXL/D3AahwcqNxuMB5b4X/REczz79ha2vjrtFocJvj2I98/796y9bivv0w2mTPWJ+Fx1/
++Z+6d9728bc//vAj38/h1bWupzRb2u0dyvOO3Wu3qFAqU8Qh3aA7oMHwIudIQ15oAdWKPl3LRlw
u+OSsDq2Vg4esOOzTE4+do94oZW8woXjzHHV2dtgDhHTk+DX08JnzwhD4lre/m5ycCmFdv6ZIrzM
Vj1YX9onor3KyoRiZkrpSHk5EpSSn82SWVUSLZWTNUSLFUlCKZy0bGCdkHgxe1KaIHqs5UozMZqT
hhPJ6NVurPLLy2PDSCTYM5LxgkGDjn7/xtVRZPCB4WhGGylR7JgXNWaYGyaQCTdNqtNKSyaTyG2H
6TBGraaGrXUNFJTmV3BYQRQCJN1cv0rlLlzUnsSH3MeGwfjgcky+V75IM3U+iRISiAw6RW/0BdVo
94anK0nNCxoa5WFKht83nKRocX/Pr6vycq3hzeGm73b7n4OEjd6yurj7wg9/+tvUXvOAFwTt+4Ie
zPWN9lhw/95PvPXnnJ2/9b3nLOQlDgd5LyAbW667LIp+fX+TwKpYFMg4iWr+8SqNgSI1GQ42EsWf
b2dmkaukgHTnxYor4tR/86G3UGiF0dGjYb1G1nKeVg0fEYG6/5e/o5le9jr76jf9SGPGB8wUeVzw
qvAmv8mJzUQwCI2QIIo2pWjiHxjBoVHVNW4HvQQkq+V2c6fzO0zOpigkURgevozynMiBD690IO3+
WaQyRwkQYurljikSjucv6AAM2VWFr7oZXU+3wCc63vRnIQT1PTdKocNpQfE+oYKXK8JNYDZlLnjo
lcxM1AFsNE2hRarSSUpoga5fP+rEPf4S6bJgwPDwXchyCGzYsCYEtDu0dP0+5oiM913KlRpvb2xy
xlOWe94c9qmRlQXTlfEedhz96GE1s23SKfiGf4889b3vuifPnz78iHI8/4/n+radPPfqXpKag9oz
1mT5+9r0/+v23fPhDP8uL3h+nA9q3/yBdPH9JaCrnlpY5VA1ojY2zVm9Sayeg9dU1KYQsrxygkMP
ZSrFIr7rpBB3at0zNxSXaXFunj992G40Sn7Z3tqm1vk5Hrz5GhXKDLp89DbJO+o7vfTc1ltgjwWh
y+VkIC8wtFhFaNMLnqxkF0bIQhUZTsShgmBsJJ0JlTM8AfA4j1uXgGauo8s+GhMEq5LRl4coYW6K
Hy7W+KjYEkbNACm1kM6A/WjowmJimUpI21U/eTM1jJ2V8DcacCkdxJsLF0ywVea6iGJ8aqgZlgAg
7SlV/VTR5dNUY92A6iWPKMK/iKsZQAT/z4x+/RRGzaT1XiSrsWGREWlubvNnZ4mEXl+Zo8fhxevT
hz9DRwwdoY3uHOOQVqhgoxUMpIQgmnF9HItCMiSVsrpz/2kaaFNvbLfBfzfN9Os5/i1dsbm6+/id
/8J2325b1B+/5hV8P9oz1GTp+9Rd//kc+ffttP9vv9ahYKgmEbjAeUBAOZMokDHpShMHiALMePCt
aLditwQ+0PF+iV7zwBF178npZZH/7oY/QPfc+RKXqEnVbpyUH/YqvfZ30P7dWz9D+g0fp9W/+JmG
c99BLFGW3XeNBI98r1gTwQIknIXBKCnyQKak4NTEDjyve0ZFQmrReDM5TnVum5eX9wMequdWMZlK
QqMoqI7WVQUyBv+ZUbiOe6eXAtxoQS7ZMQSiJE+b3rVzzUmocfyGlbKjTPcGYKstpXRsRWE4TTSy
uK8CplpZACKzH9Wbq8ShEGUoIy9Bka6IGh0/PX3e2NujUqSd0iK+iB4ev0+bnDTnCQS8XraBgEvH
nHlCXN8lrrnshbaxeomuPH6Vz5y/yz4cUYridFOoLITOY/VHAq9frsvGlouKe2Lxx2Pmc1+Rdpen
5pWPtXv/1URD+H//6n3/9add3f+h97/+LtT1jfRqPP/r9/3LD7R/76E/1Wm0xRoHKcXh3gXPJer0
qi3pjY4sKnAvhuHzpHP+8yYukRBEbE2opLzh+kE6+6AYhJvurv/0onblwWZTYet0WHTx6XCqxn/n
kLTQ/16TXfv0b6YUvfQVNgq6wxMP92Y4vEy8wONsxJaeiNBCZhwwVlcloVqXFws00bYriNQqUsjh
aOQDYJ1PUj0nHr72O7rn949rHKdCDGAsoWqBxI4ZkkUW7UhsyHqfeTCnLqcqVeHMpZHGIWb/25VR
eOSqhr2rF6F7PTKDK1EoAqFjFipxNvHMq9xg6qpTtjsGRrvgKI4WlBtzlf13pRnUZx6duvZ29ImZ
uObuOeaOx9PCgpXRpZfgcI4N8Qef5b1DijTc+fZoq5QL1e21aXmzS/pUFmeXF/Gs44QB7okLx3vY
WDbsddS+zWEJph99rMjFsRDHbly9XecMoLi7tW+CN/CVW7L7xbW96/WX2yt/LYfInfv2//GGyZ6z
/i4+PfehDP8a7sBXGCkwQc2iGBQMP2u8Pqc75qON0xRByhZzSNjUEd0TVnE1veM3N9KIXv4QefPQ
x+oM/+mPyiw06fOQ450pFETt+5IF75byvfNlL6U3f/DbOkwoyqmZYvhiqWvCWVElR7VRGN1SKZn5
JFpbi3CXpO6ZT1kA9ZWYZecWvIgs/msEK4R2LpZp4VAAfLMvYlXjEaBty3MwRozGMRKrAwgSBB3J
GzJDqim9iJpIX+odPUmHfNeTXaqLnKgUh8AojnzXMGUY/1ZIYsrlIIYpmUcNUsEkwxXGmP9+uWLP
icjIE+yu6qZY968t+6o7bZegd90Z6xxNJCiRUR4huszeGbi02BIvvb6vXp+1WQo0KP2p5snNVvj8
T4VFyXVeoTYtFFTFkOozHjVUqCHyuIKSddg91KBV9ZJm9zfkvNpjhsF/h8LjSt/t/yV+73/LG15z
hc367nys88Vt/+P50z1ifckP9oDvo9l6fce4SS/6kjMAU3iCVE61fusze8Zi0BDrbm3Tt1VfTmHd
lzxjTN77ha+nYNVfTn/zJn9Dtn7qfcsUS1ZtN+eOfOn2W+p2uLLx3fOd30Ute+nJenKEUVWADDha
3FmSyLBWigljbSC2l5cqeHDhf0pVUpet6hecDgslQrRBFLGhK+BYDzN7aFC6kYNjhRWnLIkadV4l
GcVhp5xUToUAPFYBfxH+1nKMYlRa3Mvn1pcUjVLv6Rgp51UacnxfMprpPIlAVq/BaCyiDonTWCtJ
9VTWXSmo2V7CFphp2hxfVvE9T8AM4mxCaW6JIZ+kKtko/7r/7PsUNxZsaJ5SSe0aRQjslhv6bhbH
0Ygcc9cQapWWkXRoHAa1ttIQOxrANEbZyeaPM5TxBSmXwzBEGoib8XoGEyaat8mbZjCTPTqnb6Uu
4nHD+KxQ5iZnn98jz+lkKx6M7B4NB71++/mvuYsP9IX6c+50/+UC2Z6xPwXFx9eKBIAj8ELOYQNF
```

```
YpshNYPdMJawzJbza2lwXycEjx45RIe9Sndf6a171lbSwvEQf+NsP0aWtMR04fh0VyhU6d/YSrV6
+KAZ7ZKVB7/qBn6R9R46qciwbnyCBrtCQy7RGapoEkre6BZ9cNnpUWg14p0gpu6XZVLFtt2qUCfB
A04iaiml/PBoQn4BG/R2prvpYkK5WOEcRB5y8/z97XwJt2VmVuc9w77tvqCGpSqUqc0IIpDKQoAt
FUEREY6OgtLpaBAUVFIwCYUEnC8WBOLVTu7BFRWjtdliCtiyW7QCiIiZkHitzyFippMZXb7rTmfr
/vr33f86t0L1W9zL0SoUKtap479W9557z7/n7vh2urU/kklGQCS908bOUEX5+xymydM5F4c8zSa+
D4RUrK7Kxthpq9vvklHMukIZAipzRCCMYzwOryvegm3EiXS+ncSVk0nQAF8ab9b9nvv8VcMeyUJZ
P+Hx7brouGGSms9tKx1VoCiX1METajBjlYlqTeF+ZY4MBglmkFEAFaMDh4EJ1O/k0OJahQRyFpYt
eeY8dYkWCpXb/242B6FmgCchnl3CfsxRpASz1lvDCW7KsPj2cp1cF4z8SUuXPhkzgA8Fwn/r9P/v
L5ivG+v/46/D+Ayd6BNV1hTqHRFewrlLTs03YpFlbWw7p1BbpN5vkG155qZx0yk75m7/9tNz32H7
C2zaGpew/eAdX3W9dWpRveNlL5Ed+9HLO+nTDqeJ2HRmko5mU1DBucQsHBYAk1HxknjRjPfwmQtb
YPLXJNemDXq9MK6KQVMOopqES9xtebzoakxg+HyIJRkCMmrUupNKUtFRZFGtWURw4PMG54HC2Xfw
K2XTK2WpIKRUXwz0ZMbrh2tdDhvFkOKgnnXYOpU98ZWT0QYnORuvE9uHUpYmfpZzrlpa90PkwW7A
IhiabtNrD3LKe99lAuv2220JNantisUoSguLBqYDXC3AEYIeTkOI2HQog9+RaSstoz9wj4rRM21j
NE/t01bEkTLWpMUVkVTWTbcBXDkfrIVvp6cy6zFgm4GGRyuD/rkq3hDR/SzEpfzDPi+8ZpuPVN3z
Ht94TsqV3zs3NP/iRP/94+RVj/b8x1kPLKwUecKOrCXFY/CGLpXPTSa2SnOHQDMcjed6ZZ8kZIS3
+zKf/ST5344OyHOqiKRs1KVOo551+slx++Y/LhRddom2d4NWxQ7WsFIdL5XrUkfD4mQLaxUYjTD1
BHKdifqo7ZcQWHIOWRtX7hEybdLBIORYcaAiIQb2B2UHjzaaxjNAwAepHdDRT2zLhHqJMqNmUedN
TnaPw9AbbTiEHtRdqZWgw4X0B21NsLgx7TWe64b2P7HtMVg49KWdf9LWq6wSGDlDMkHhJmjhrdVo
eZWLqql26bGkvFW6AG8ZyZ46KdGcr692QXWDEgp+95l+ukbn5BSnXR0p2D5kBI5w1rdiMorNVQ61
N/X+OZACto7O01ZNi7yEzxQqTVm0s5aFhd5wPnatlAmmTU7sp6bF+ILeX+GpND3RubIAVzR6yXrj
OLeH9toR/f3r487o02xh+72WvXg5//52QHfz3kM2tfeiP/rj5irH+H34dObR/pZmWjERcKwgd2rJ
krYL6bjIeyuKmBQlliNTjibzwvDPk61/+1WR83HH/XgajjWDAC/OL3D7+zS/7Knnne95rC3vVi4O
UjddPOrUnoijnhgDTFzr7gxRLH/VTMIpyvE6qWINuM5g1KaRul3iwVW0QpWLBOWtDGltI76DQgOZ
Jo2il4fKyHN2/19QKJY50OMKpWUmyIQUlCaT/g5D+n/L1r5deuHbuoQl1eRVSV3RF1YGF64ZsTDD
y8WiNMMHJ+oo8cOu/yrkXf40MFjbR2dVJu8DZTIBZxf92CXiTRkMHhS5NPfOo+XVc68En94WS5QD
pdhrlCn5GUPi4vqMGbKJkDUvwh9EFOT/u5a18jQE2tLmktT9+Hq/H+hzOggu2JAJE/N9w9SW4wkY
ZVNKBSuuUtbAnUFqpUtUqdt7rJ0yp3VFSd7mut2RNsmVaTXeFUuQ3Q2b3s8MQBd70um+/MWRh73/
eeS948H0/98FnPOo+6zSYhmvr72STj1IhmvaurhyWTYub9SFUIGdLSGuX5IStC/LSF58njz26X66
77UE5uj6WtdWjcurOk+X0k5bkZ/7jT8p7r7pKBgPdPUo9o6o2RftUSdrlWBE6ENVGPbm4SRa3nih
LJ24PxrpElXc0e9AY6gXjBB+ULBymi6rakA/UmJA+Y70j0vjJxhqVBElVw3rDEA0f3nNDiCRzgmV
MvVz3r3KOmal0KAES7HLqDtVtl74q1Mub6BxwENeWn5JssCB7770xvP6qPHrnNfHQw6nhd5YvyGS
4IQ/fdr0Mg+HWnqo3tXaZReLBt+GO/p1RNY0RDh221OCNXht6YwnO89abbg6JRsZRS2nC2t6UWgj
OaJPdE5WhU0MFbWFu0NMU2NwGRloYiyHlRW2gBALFD8Ngq6R93cacK94PURXosliOiDpIX0yN51w
0EmtaoNlA3YMjwPUXALNULcySXyvQHCt606rePhmNTx+Px69fWV2+7u49tz125U+8/fq7br01+4q
x2q/3Xf6OU4vx5J2V3TzL1oSgICzx7c2RegUvDF7pmbtCLbftZLn1rkfk4KFlCmdv3rwkX3PJOXL
1r/yqvOwbv9mUFGpiWxsbs6TW8SWyCF1ZpIw9lQCFN0dqJ7mOL8hnDVEcurTo0iKS9RcWKb3Zn1t
kTQsGThb+XiY9Rn4cFERpHCaN3KWsLx8kXrkPAe6ksfpQ2H3VFE0PJbeNhzp0sOsM2XT6ucodZdO
4iE7mhJ2ncUnztlPO4riHRhYOOupdXE8/n2dNu/euW2T18H41RBbfeujTug2pnnKm1vQBtDIx2CD
VJHRbs+oFI1VLdKZ6W6hXAU/0g1+A6I7UHfU/HE9ay8LcQE7cslkWF+YYUXsc5fT0/S3lRroOOVN
l/ujcGs0wNBjxJxxsXVVWZ2ujKrV6H2VSbUZcdQEeeI3gmIpClTZ8Kx+wycr9bVT6NdNUnBMH+3w
QFOAzE43U4TNuGQ0nuw7u3feSP/zdD3/2njvvSL5irOHX2urqz42mk3mvMSrz2LjxAIUDEA61hcl
ESdIX794te/bsYbo72hjKQp7KG77nu+Rtl18pm7ZsZYeVqQ70j+qMKSSiJz02tqAhGvXnLZrkPKw
bo1Wij7gqIvw8IiciKldGAN8aUlJQ3TI0goAkynocUYyDQU83lrXezXvsYroiA4z38KNfZLcXXdI
5pK7WYCHg14yHs9BaeaPbdr80pGzKloGjwmwzD/UyjHbTtjPD30P0MmUKjfQKV+yHOnNurhcygiW
exv0P7pEjTz1G44hjoMZmqKlBCjONpExDo9xLHXs+NIJKU9y6l1LE7c4779Qo5wguOLbMRje16hW
ztx2+vik4tx0nniAnbt7EiKoGl+r75ZkR3xv+O7xeYfhk/MmZcaZNrjgicyfT9BTtlbR1rGZMZZs
hiGZpmpFZhE6s4VaJtRlFtZfpuGz3rcFBcT34PQ73+cDhA1/3Jx/76FXPeWP9we/+zouC0b2ZnrE
yLmXqNUfwtCFScGFvOAJ79+4NqW1O5MveQ4dlLaTJFzzvLPmpD/yUvPrbX6c1jBYpOjPkCkXlfDI
NLqzFj1TPQRDBaO69e4986INXscva2KLiyTTUgkhn2e1tdK0iR6oFmy3QBB6vHqaheiqY2FJjLKP
Ce06LkSw/+UUaUy8YXb+nIH9Q69JGjbG2xg8+/9zmbTJ30k4pRhu66mKq2OKFE0/ma0NmtAoRH8a
TWzoNUAL+3qNsywIJDxkdSV/2P/KALO9/ohVhS7XzzGhVaR2axE5wK9YmndTXIxa0pY7s3y9PPPE
kI2RlhxkGg5QSjZ2UTeyEdaPXpWgkDYLzWsA29DyPo6A4IrL3QqSeTEtvBHHHDtPUpDGnkBndT8s
FFKPsxFeNRds0dp85amqsJxHufWXd76jgmJTkHzsJn918VOq1xFm0y+LQaCfT3r59e9/9m7/wwe3
P6QbTxurGL0GPl40KayMgdY1/hgeFcQ27nOEBLoQIVYeUanjogHzTt7xGXnHZ65ge4/A5oF4XIGM
4Hx58SIGSVGsidF3TxpcRJ/J4OHh//zeflk/99V+ze7zt139Tfux97wOcgSklZU8SleTEvlHUt5j
PTlZXZDRaYWSYW9jMKOtpL8YbfNjhoK/ufYz16GJ/SSVPjPkCBQhIrwD9B8I44kfTyziKmAvRHPt
bWNdFBk9FXC5TNjgCbjjPmC2kaRFq4UWpsorXWXG+qdh2pKv7HriLc8ztp57dRhw07hoQARTs33T
```

qWVeNQDOp5hY7XQ8Z3IPcdP11wRAzRqMcXW4bfbEZRS6usYEaPfj+frrqownGOqfZAuKaltR8Duh
yg6UjNprJ2zDK12U/GaCTJre5rTo4NiBtURfhm6nOrguSBFKKBtRJQXqivldtvfDWWdRJ3ZFhFWn
VMzL+G22Yhc9YFNsfuOfeTwIA95w01n9/2be8fDwavSbn0qZMpU5s9QM9rCiSCWnSFOnuYE6RQuW
qvOVt75Dnv/jlCgDAlnCrNcHjhLekTEpFyL2mdEjlbJTwwAMPyKf/8XPy2X/8vAzH6ChqB/K/fvx
/yPYd2+R73vIj7BLjNRpGMa1BJ1h7sbaiN5j6uX3+HDmpUwVyCFUJtWGEFYgQ9c5BDkh0hgiPD0c
CA4B+ApJqaieBNDBYYIeXyg5wPuE+lDbOSmGYZhyo6xpbB5lDqUJiPhicBfSaMh2doBYLYe6Je++
SYbjuU8+9UDOAcL+RVovzWistL8jMrexPUOfQrUWECY6qCD+G+arrRqEEaHu0tY1K2gwj80SzO4p
h/6HHFDize0GgxRQ1ojrFAUdkaUwQfcTEKIf/Kl+epXWtp8E0rKmWT0BD5ZEL3OPLMWXONJVPDb+
cJUaWaCTuFMrEVpk4n9dqbNTB66P1C6/4oR/a9Rsf+9iTzzljnQxHVxP0gJOd61y16owV8EiZClZ
6LMDK6KcDOf+Sl8vC9jPoKSuCy+eoC8R0CQ84qYlnRaSehod7YN8T8sWHHpQHH7xfntj7lNx/3yN
yNKSxG0NbhZjoJnCc3w99+A9l84nb5NWveY3Mb9psjqPi+onh8iFtjoT61eeAUgWTCwcs7+c8SA0
fdh68+7qsH3iSGUHu6V9Z2DY3RGBqBvIQ9VJN4QbbdnFOqygeZf+AbO/4YxXvLhgZLffmjJMqhag
Fw8GcZhXvW5EUMcWEUsVjd98pTz32sLzoFf8uBOcea3EpxTq+2sTxyJqwjOixOUQQRaO142233mg
/3wL/U9/V411hj1geHC1qewMqMckawAybAmXOiKkmsc0w4DjaUmK+tTFMzULTVRhqhfUdecOMBR3
0qRkqG1d48iRQWJ1KvamC8FFgt8HptSvujJI0rU5snSYcVe6Zg3Wlw9+3rG2s/jbizHPKWL/zm15
52XQ8eUVBGpSOZQjeTprY+i/qmkgZPCJ8f9eOrfJD73g3lz5JqjImNZ8mhuNT1rT79j4u9913nzz
2+JOEGu4LqejOnTtlvLEuC8H4oJiwztUShXFGFblTNyqQvRGcx8/+wq/KZz7zGbn83T8hL7zgxTJ
cPSqj1SMhAx6wI8pIlyh+WEKtiMgO8TCAIIAA6oUTsrrvMXZA0QBKyG0tyHNFLVaSWpfYTBN6+gX
XQy6eclZ4reCQ5jfTeEAyQPR2pQqltFUGGsntIGGPaoi6dHRav1WI8IWmmCgv4BDXhuvyd3/1Sfn
cF26Qt19xFXmj5NyaqmHiKXKqMqXs7JY6okIk3vfIg7L3qRVesy5jrmPTJpO2Rm1MMdHRR90VIHn
u+OssYoW1ZKjZE0gMHJFZ82tqNTO1kBMH8CeM6FnW4pinoZ7OLcPJKEZet+spjbjgk4C0ERuJtYq
nGj3rCB/tItuIvMpbAkO4ppde8aNvzX7j9z5SPWeMdTgcXu0PjLxRo30hDSlQD3Hmpoihhp26TN5
31fvkohd/tWysHJDD990tt91+gzzy+H65+549IWIeUBQMVtdvQJhL1fS2bt7GmnJx6zYOyJ94/An
KZ3KBU6oa9ogchM0Bd1qHFDHUT5//wm3yvLP/VnbuOp1REukpRzfACaOTGV5jMjxKg4ChqgGJONZ
mtHJQ+oNeqH0HynoREyRj8LcxgS0sTtK+7Lzk62Tr2bu5QwbGv3Z0f3Au25h+FSuHZX7zyVGLycE
IVHUgyyaJTS6+Pg+yGgicIbKChx7eK7c+uI+/H310r1z+nvfK88/frfe8h7S8ijDFxhc0A93VVwL
DPXvuieQAysDAnpqkjZzS6k8l0o594tY7J/NjTgrcr3XsES0VlZR0iO6N1Y66vhKvxXGM1argHmf
UPg5Zw2gsOWR2uFy6cU10mh3xzQ0fb7wOVq1NFaOpNIRQsJwoTGYHNEJ3Pl2xOnMuC+Fanh++de9
zwli/45Wv/K5xMf0qb2Aofr1pwez0cSGVNHVA3DZ0hX/1135LFucHsm1R5KUvuTSktg/L48H4II9
y5jnPIxxxGmq8pU3CmenK0cOytDAvCwt9WR1XsveJ/eHhBiNemGNTw+guSksjTC2NhwRe/ROf/JS
MKgixZfJ9b3ijzC0msRZOe3PUDC7W1ySZSzRVRI0rGt2AgmL6yxFFRSMj/BBI80oZLeYqmE2c9nW
XhSC9xDppvLYsg8UTJAufdf2JR0K9viqDTTtiulqXKr/i6CBqC1vKmzLTCwlxGj5jMw119hgq+PL
Za2+WArVnKCmuu+1eue/HL5cr3nW5vOrbXqsHFK9j0jQKTVQ6HZtf4Wt33H47a9XSm0BOVesgkmb
GK04kSFBD9mZGLKnJVZWFprYsHjBCgXGhy53aJgLDAJNAUSu4BSfbG4RY4AxUVI/k+EZcjoOZWaI
MJzxHIiIziZpXqrLhOzUrwjTYK+kgqzxqp96ctNFQuLYtZTl9f/ixNx33o5vv/rZXZ2VZXg3heUT
OgvKXuqrBW/k8NAZVw+Hg/DX878jRdXnk0Sfl3kcOyjXX38ltayeedCL3x7B5j7WheHDBaPbvfYh
YVCz0xYN9KhgqSM7zS/McBXmN4i16YmiNgqX494QKBx//07+Uj/3JX8r73nMFa99yqsqEGOsgfex
v2hxVIdhgsnRpLkRhboRL6ojCUVhPrusqXJs3/JoPNfJg0xbjxVak7yHalcORrB49IJt3ncMGk2o
Rd5Yd28GqTTeJXXBbwIw0FvNmgElu3XOfHDyyElM5fMLDRzfk5z/46/IHv/Mh9gLECe3WVIHMC8t
siACE93zgvvvZreVunY4xdqNmTCs76SVnwOHaERkz2+heSbs7lkAGS4cxkkoT5/pW/DlkO0VduQe
PiCVccxMiLCYBOi9OO3RAbSjh3+eJK1m0tEN9vs1MuuvnwNP2JCp6tNhpRZnxXL7sOTFnnayuv6E
op7vZ6TWlv8ogS6ndpMoedy0mXlBbakfPju1rudx5/yPywCP7pDfYyt2n0+l6iEBr9Jkgm5993m7
ZGlLfe/bcG+rXg7IxmlCtgGwV25qmzZ2MHlsPXGODdpNkQXRHqj4t5Pa7H5QrfvIKefD+u5lWjkM
dOz56KPx5iMbDfwWAQq51IBBPTMsNHaWHF6l3E1c7Onh905nPjyB7vk5/IaTeQz04kxVtBjnHNKr
l20JlwyfTYfjIS1TBH+D6jeGa3Hj7Hda40wMtxlvFffjjP/24vP8975anQoaim+8wOjF5G2aVSXi
N1ZA6P87X7WGS2gH/dyNq/FrjihMpu79jOANL071bXLEpNGa9qim0Np3gYOva561NJHP4vYIRw/A
BAJmze8w+m7SkAP2tpQJ+Hs9YBe6sSw3HYOAUyt0wktfxuftncSfhRq51LwNK/t9+/3fT49pYv/0
bvj4LHvEDUHooO3KaHBOQnaGP3lXjOYowQevBXIiIhuPEoqP5hRNCvXpIrrn2dtlz98OS97bIjl2
7KE06CjXnU3v3yj333yvjELFRG28JuTGihQMUegTvKx1M+aXtYWKapOvE49wRSJgHHntCLv/xd8k
n/uzPQ4q6SIQUonttTZ/KRgQ8tGwAlSbAXcWDkNuhcugb3m/7ORe0HWBJCMQXq61Ou+Abmc4pOtA
k9cN7IvWvrIZjJKJ6S6rRObwv0t/JeCy33n6fHF0ba0Or0aXNOByAP+IflWUq19y0R97+Iz8sf/8
/P6WjkrQ2eKF2q++942YZV03sGEfQiUcqJyak6Uwa3BvMMTIDBYSSgMJwYAASi6sZU03hNvM3IZu
oTG7VexmentLIOeZRQ6USYuIrPZrYZXZD00CcMBvQDrfMADJaTqzVsibZGpleks6ANhzrbOSB+fX
VtfnjumYNnv5Nk6o+VxsLxF8q3KsON6bM6FkdXx5vlMHx5oJxjYInpmctlS6GFAjLkMbL63I4/Aa
kD80HPzjbt59C7whNYW24VCRB4yETi4sDV2k0TRXXxr9oa99aJ+Fns8Y6lujyrk/kw3/wh9TQ/f6
3/CgPCQ6210uoW4EN1jSqXVFBMgJHFD7qqDluAkFg29kX8BAnVDtUAwY4IkZPT3fL0sYQlTWYah/
jh/fS6IF7S0JBSKEPHz4ot9z1QBzwR6CB1WOZRXnc0wOHV+UXP/jLsue22+XH3nWFLICaZ3LiX3z

wUSPW19wCkHS6vHp9FTvLbUMmPK/+HNNcQEIHg0Gs/dgswggL+2MdgN8kfHa9TCOXc5l1/w9ExXX
OPQ1OOI+GqiJw3NNjCDBGyrqJkMqynvKZJnG7uzewDJTdtJ8/7YoQNLqSRLnK2hDEs+slPf/svZD
tbAGm57iMrD/93vdk4UO+H1EINR9V6esibi6jAkA9iyZBvYpadH5uTh9eoSBvztyCh52ElHZ+YYF
aSmBvANlUWzcZad+BAwfk6NGjbDpBvhShT9MYjXSZTgSUjlUTWCi5eVo3tDIYUGUbwR2sAfmSj37
0T+TWG77ALqLXrGLNpTJpjqGCVTzkPpOEs/D6bQeACj0F7Yt1Vx0/m1rql3QgieRy1pXtuGki6ss
xvRXJ7iOykK6/6U5ZWd1oGSgdp5EYsshXYgDc0YRo/Vef+nu58l0/KfsP7ONMGNf+2KMPU0CuMf2
J2lFOlnInpmOMLIkRKlcV/5WVNQNu5LGJxQ11cNaou1GzkwpXshGH/3Iaa1srkuMb/oRT1tfpzTi
K+KfpO7fRsIx1qn+N9WvdmPhbGiOoj5A8M8iyufi1pNMX8N5A1dS9IwcP7jpu0+C1tbU3hQ99Lg4
X0tKKH17TnLLsYFDjnl9t3+NYLi4u6N5Q0xLizc9Tinyth3QP0pdYe1GFaLu2MQo10pR/NlQx0BU
QXWhZJD1Hg0q1+aiVjaY7NiJhimlNkzppebDrwVl86i/+Qg+8SXV2my7laCMaJA9zBy9L52ANoZN
3X6pLp7z/5ARsu64WtFASDOERp3kaKVWJAcAsQxlw394n5YY771XAPRyfKOger+Pvo3WrzW3jfcl
Yn//EW98qt918bXhWjTz00ENESdX2GmJsFzqwpDUQdHRZP/ZyGhcifM/w0F6nQgyPErJc12F7acN
//Z6m3O7oartfyFzwWsRA94SZk9eexEdbXe0bBirrRwAgktnXCiOGkDBQTWOTSt+jbSqRRFDYImo
4kqy9T5GxVSj+/PCBg6cel8b6tje+Idv3+ONXKqKLSAR2ALHxDB7WaxOvF2IUqPShoC+Ih8tmjyv
QN7UBux0rU5KATiBAbCCFw5Jr/dX9lXDPy6xxOXqlawQehavY5u+kpOFZXn/DtbJ8aH+km3mrhRG
0MmidpYmtg1CYG5g/nAOfrfVqEgEFbSRwrKx69opjiBIk9WbaRhe7tMpmkZPhkFH18zfcTM1eRfC
o6qED3l0qxdNARx4pFK/Hz/bEoVW58sqfkb/4+CfkgYcelwL3F+OWRuVTVda4remTzqwX1xScM3W
YtTfgYmtKrZsYztlZN9z5avKsxPbaqkyCJ4LzQQOwnyJN7ptEqjkAa4h5Bzg+y1T1lpmFlc1Mt7d
qlJbnZ83PHUZIPstlSm29hcI7KR3QB74fAkTvuKxZn3z8sTdNp+ULiC7BoJ6H10S46NFKDuLnkDK
FnwlVKKU3wY0cDBboiWkA6OrBCIK3pWEmbdsdKawO03EzNd1lRBWlYlFgtNE40piIN7XN0oicm4G
dlaIjiqTKIsCA8DrJxOnbeTYv9aSITJWY+uJ9JuumeqizQ35GzDBTTSPzcFE7XnghQRZizkUzYVX
LF4s6kZeLJlZR6kYAp6/VdYvng/MbjxhVH374Ybn/4cf1+7VDF9OZVM9npIgYcIIpZ7Q6X3Z1xOG
4kt/5Lx/hoU8ZWRtb6mxjI4NDcB5r6TQiIQxVMchZTH8ZtcqK8E/NpHSumlqnV39OJUiRTqccPSn
hHGMwYJ45s7b7i3+f2mtr4yqJTr4wWCKlgCpVHVHAiG/eaz9DWVoDiuoUGadtng3EURBBcmkcSqF
mDZ9h53FnrK//1ldlK+sbVyJ9cH2gXHcAs3Ez5ewuZW2aI7JkuUU91fJBqjMdVRHkz7THxixoTEX
CMAW40pi68ucaNGr67IoiYlXSImPUyGs+OE2ttQsMkYqq1hrRUUkejblpPMEmOU05X7j7wlB/12S
oqIq/UsaY3o2HfNhQLYAqH+rjOk2tgaS5z66LvpbGic+hN6WOc8rGuq7swMZSodRVjN2uZ0OVJNb
wo/G6rK4sy+euu4XKgYopTtssAdeZK6mbJmyHVsHrdm1N5J4YeiiJiCvtI9RxxMZ9sOEn+5gd1xA
6yzRyTnSm2rNGm88zveRhVKtUNwlb/YDv7Y5+/DemBhBh89TXv8/nlLV6Tf4LkVHF1mvTemp8Lhr
fV0H5w5iAamah6TUcBK6bDSzrT6hzahQJlcyMqDaOO2OdjqZvCjXHC6DwMC6mHMFI0VMAfDjgGZk
lWgdOwo2aJ6IpIehacZ5i+FBNi5Vx0dOWuzV0akMgqeKCGiF4naosMTFInpKjlFHRxI6zmoaq5te
MhHjPSQeapsSAXqIsEY2pFR3txRdewFQbndfB0maxrceMFk04EKSN0YCb2AVG9EKFtbhtu2w940K
jkIFvm7QUwaajouglgqs+dqRW1LAr4o7L4Bw2jq7K/fffL488sV+jcWXxr/FassfX8ToM0d5Aeor
5JeVMI2VSKX2P0a6srByRqEAoJqOSO0w0U5Dw+tF13kvWk53qg88wGqq7m0p6fd8r1BA8USveMTi
eIfW3YPBiMjPqECvuNcqMmK6GI+wwK/Df72Ea+xygCHr01D5CEvsJHDFZqt7FAYt1kJGN8TlABrI
fR1NIp9aOK2N9+1t+IAtpyHtJAgfDYrzBfSipUbHycPNKahhVWhONSsE2lYyylOH7c1jcWxk4Ivw
M1O6rPKa/SW1euCLEMx5g1DhEvowLHQWkHoUy0yNSzVtGlbRvCn5V7KrSO9tAnDSCRuIMVSwi4bl
d/KKvYoSFpAtEzoQInHBwJmNGfygoZKUOQBBB6srS6FCj7TjvRSbtkhi6qZqVJ0XGQLqgRwUd8Es
HZROH96FM2FhfZVS95qY7eGC1k5m2kp7Gi2WEypQuWHs9C8OFOgauWRRoEaOc3wefZNGJ1TxiCRX
+oVKh+sZAFRHnnQ84ivE6r5VKgaKjzU6BOEqVSAeGDyIYyhjco2mhuGFlLKWdFsxsberNtlb3uIm
osLiexHbMYo1mjjFb3o9jH98m0BU3byl3iYrAhXvBMoZw0bb7HL731HFlrMVo+trwx+7KWv5oJq2
trcr84hLHK9QkYiMoYZo4aaaSFQkjAOwODx2dQAD5a9vajUNN1fq0ZxxR1EsaeevaVyQ2No808nj
tHliXHuOFsqzR2hZGGzu5ipJq6ibiT7WebDgTTD0qh//OPH2n7Dj1ZBprvz9okTxQI9xQ9X/ODXP
tfkMELkkru16Rk15wIUH/yCxqK4c8PXUCN65VO+dqqImPTNxgARLAGAyz5o2hPPDgI/LEk0e0xky
0RheLQpGonWWmBoiyA0gng3N6A6UuZ0YgmpFo1E/MCLQGVnhmEyvXNDyroUav3KRFjRmD1ZwkJFS
WijqQIa1jhGvioFMV/wF8iOobVr4kNud0Q3XKnBsYyxQXcWv0HCAy6/hIzFAVteS1tNemeL3c0Wf
x/6eddLlFSZFvl6UHjytj3Riuv1db7OEmpZqKjKYT8k9Tazo0iQ7Eq0IBCpNws+cTTaFy83aVAdV
z0xLOLYKpZGbKZlNiY4SebQDXTd+Z0tDCAwCrp6hLjvY1nctMuSGzw6n90MrV9g2xguiPqE/lpSa
xtcSNXHLRxfrAoXxgspw6P0+4oa5Hwjo+n2oRp5pMEHfRCz+/5dQX6Gdri9RoqCpMrUAMn0uyH+R
0M1uTwZ8rp4QUIqp+/gvXx06rd2kzi6jsA6RpNFTpdL55TxJ1FqaTpg0z637ywsUghBZitaQQo9C
hqbTCe9rP8nZXTiNxnw8hkFFDuNKZNpZ68ZnlBojA+SgthU6tjsyj2B0jP+GptcnS6HirsmXQU1P
IID4cUMd+L5IIXGzNO3JNpxPuBloa26Y0vrA34Rzp5p1gnJnwHsvHjbH+8H/43leEWu6lupgoiwR
sRMjR+kjy+VA/jWHAEyJOUY8AgoYoWZQTrmM4dqboKQ4eBnHCEMTSJFkXQ1n3jjPypkcFdz1cNQ2
a6Z7r2FpnWPeXJmT5VAY0SBmB1Zu6hq5768q88cUX7TYthNRWWphsC65zvMYfTnOtu7PaIHOJzla
Xduy0reQmoJ0lNkutbLFy1daqGNWA5NB0xlqVdrIdADFeXZU9d94re/cfjoewMgpbVOePkivNDIz
Osdi19Q28aVNHB6Md50SSDn/VOsCidDU8N+xXRTQECk0Xb7WgkGlwIBNLbRWV1Na0NMJMYvMK0RL
pb5Y1kenSWHcaDSfM1kFtpNFXytfl+sta5+hzc/NcU4LPC5yxG1gmLR46lVlYZJe72iUgeJ2LUVJ

```
mUR2BB+uXwo+tHzfGurG6diVSWALnIS2Z23gFKnZNIX2xRU9VxpRYGyAqZwIM8FLwupPp6OkvbLt
Aa0tz5BjUkxqsNoG0G2rcVRPBooYwamhSo2prCCURjR4bSzK7fMxHBniAm+YHcv4LL4iIIIp/26Z
zAhDQHEG9ihWOkJsBiTtV0a8sHM6lk05rxclqayzFxlG7OoSyMpWpH86AILSWKqlTvE6U1j9fd2O
ILJmlyarWV5rMZ1d/NzVVCq9lOzzNmMrTgTTJlzRsr19ZEhjZ+8iR/TQ6B9srXldaaVCkv4VitCM
7qa39VOk/PC/0GJyrmuc96x3YZvakMgogiPWVSY5WPFuooQYhk4GcC2R0ACt1h8JswcZIlfCmu/S
r6ZybWdCMNv9U3CAzzDozviQdhdecHBegiDe+9jsuCUZ6Ge4Dd5+YNAgK+saIy3iAmGc6TG1q3U4
dqkOWozYQvA7PMxtB9NIOra1uB9bdEYHYaydWY3q64+CI1EHeYsN4q7+yuGiqnnlgXXA4Htjzzzl
DBgvzlP1IjbkRwfr4WSy2yueMXWMskES/C3PetOPUzpxURz9iWYRHVO/m0lA78Ls6Nk9K4qOHw3W
57fa75NDKGutNprxZEoEfXbUGJ4V75u0pX/fQRkikIax8XBHxxKIG5jDC9bUNyAbT4HRc06N6RZx
TwtAKrduTqiUvpB3YJe49BAFG5Vg1kNMOWbx2xYuKf0f9CzUPRHIu9ZrLZNNggRvW+/Ma2R2c311
f2VLdsjjOaQXMq9l7QNJAEnmxDoaIoIssK//zx/6oPi6MdW1l9cr1OTAYaqEHmZlqxcYROnLcWh5
S4KyfcZiOCOKqEXqz1NCmTeX07HZhlXlInOQ+AOBp30AR3nmsSH9C80k9YR7naYaoNf6qmNJhM6M
GMIM5rVtkTiXtoqTdLzwvNijc4COwXOFEbDxz2O+Ln7I0HoRNO89QwLnpGTVW24mlorgfRHnhz0q
V/V1jOLGfQVQt1tdl5egR+cLNdxIW2CSldXZL/XxRC3jW4GobUWikyiL/ta7rFv1l4HV3kl1cc2p
NK/z/5VCrcvSRt19zEEbsVjuYI1WCe27fx2gtMaBDOZrIwmBe5kBusI5t1XFckCmF4x9PJqxP4ZD
m5xdZLkHLGZ+HHeWymklvnTHjxtjlsHbT39gBtnoZRo8udW0puOOJ7WdG/5b28v8tDX7HD77p3PX
hxndT8NkOMregmEclv7GsWcf0JsGL9jIjoIcbNw2pTa+ReVKrpnH9X2OND6R1rj0bwdvcLp6q4Fa
joxHp8j7tKuhtdYWpRggcEG4Tz+IDBWqJQAHpGLFFkzYtErn4gudHkjYiQZLMKt17lC44JlHkKgA
aSL96i/OycOKp7WIog/1VcaMc3YnqCXNloo1sbGTkAPhyolH1llsRVVtieaKMaTbGnOWSdiRVvB4
jUslqWPYCuPcnb6NwrSUGu8dWT4o10FJTdHjiqQO2NEzhk91sBCAYABrGG+OZ5VQZ77mw4afprhL
TKyNsuEQM78VUa/YiWExhGsq47/MDNXKfj3bvvZPh0XeobOzk6DN1HBMz5HZMk3UVJtLWqVZJFTH
HzjcOZVR4rMnfHBfGOhwO31lVdTaajInnbGdzNr+CwnyuGklIexbyeT1M4bYwNS7V4yrIv9FhdKP
r6lNDm+hDMQEsLq4qOjBB85gY1tcaWWqyd1Ld4J1qrM5qPdjR20rT6vLYw/GvIernlqxsWexzmzq
GuzXXV/TaJg6ziJRMEXjmpFB+KuFx3EAnsnnX2UxCp3RAtY2YZNbTS2OKMxVT4cYWKVUm64LDPR5
tsAN87U23WAQx6l9jEp1miDyIlQIzPHKUhgRTBFI39WtiI8pJ6l3cLd0OZ6qZrIb0dzweyXwP4t2
p6SK3h92VJyDH468PcDHev+qF+2LEcaNO0vCkk5qjHtVsSrmvkIvFGGYBzSdjJfG17BkhXSV2HN3
lRGGTfj26TT2ZqVOjZKql7v73eK2iTqWFaNr6jl5a5L30j5/1xvrW7/++LeHGvxmfEjjW0ob+kdl
ilDhge/GASsLkpppaie4KrZzLWmc22qi0prSdJTiIaaIHuvJGECJrJTYTtNV/RgCw6ouRIXcRA8L
V6qg/0tXk8bleBKY3EqMovrb7vHNCbTSgkRHSaCgrjeY6PpLEIXnSQSRpCrjplDMZLbLK1j/Yz/M
T1k5e1y5oTYjh1EZCugakpMLChEuobgpR9cjKBp1J+EDam+ZCqTqyaIx5F5UbirKMeGOxFR9+jVW
qWQiZqz668W13NIiUTUBE2+XlZa0/8zTySx3q6WssIm7bmjc1uMZAKgGU38tmVAMxqyYSqbBRFVh
UjLgVXzPvLYT6NOfYyyafEagRAROV7cyRyihtcky9mmiTr5fE9L5L3/Mxjn6Wirxmr9kT4/6GbGA
Y/s2Dz3pjXT+68rbRaLQETwWPh9WH8IA9Y0k0nJ8GQyot/QnebzidRCQJ66hSU7iJTKMMZWOkbYp
3YbVGqg9L9XOTqCukRldHr9kqF6QxpYWhsFvpDJlUubT+/6t0BijTIabr+7zo/PMZOWuT4JxJHTs
1b+PKjJEdoyiphZPOsLpU1fygWUz8LtknjYqr1UYO9ENmKCDtnan+00aoV6+76TaFYSZ9pa/hfbi
DtlAyvNfinYPoUqOJ9QG88eSUQfxcAdBA7amg2IKoJJYx+w8dDg6ipvGmeRbxtT0uYJ7tqMbUONU
NfN54AkEf7YTh+igYam7XqqwcrNAcTQuL8LrECj/jKh50LLhGz4BqxZUTmw0jLdIWyZSlsdvrUqV
JMqtq8TSDFnf8orP98LpzMuC5Qb0aHM3wWW2sV737ndl4PH47xjW4KZh1rY4nqv0rAAno+kVAzEr
WKqYd29RxcxzGEQtz8/TiM905p0+JRTKrxVw93W9um8K0cDTdwKZzTAX3J7o3NZnt8npqRNU+7jh
RUGKdGMoJ8LdwVy960SV6OI0EECVavG5KJY6JMFZQQLiBwkGU37xd03A0oIxY3RhWWWesen3czWN
7e6iRXOnu2jpEm9FqMNTrbpGDR1dJto8Dfhx4NFiyjoSmI3TKsuXl+jY3ywjgoPJOFMqa1lBBQiA
lzubYwHdD/YH3OjxDjEs4tqkzfnZC9Jq6ZTpV/q4qJ+sGn6UthJBkb3JJJxHthEsEoGEBDSR2pes
4loPsDIazRWm43fAaa9hLVCsirnJ9pbytRVt1ibaByZlrkkbVw+5Z6BETrn+fSwz8kgGVlq394oc
+0jyrjfXAk0++JtzkszFXpaqc1RTciM19yC20jA801RSEc1h4aXaJK6KWOLLhHk14eEWtNKlED15
bupjF9DRV7ic8veM7PT9Kswj2TkzBUGzREeCOiPaUcUlaVTwKqZmnj9vHwvM6dcdW2XXaLlEKRkp
199k9KWIG6suRK/1NJFMdououbT7VOqNMOzUhnYeD0Dm20cXP8XUbreGAvz16ZFn+5fqb4wiC1+J
R3Oo3NdTc1Os7js74w10WS6qoByXSWyNPEu+Ga6ZBcP7cQJ586imTjRU2lnoOiI+c3jZST7lQWnH
A6RTqkylJ5r1M4YbT8dSUHyqqb4xGE4MjppSchYPHtoKpAWvEqBKcBIoK2QEDzl1j4TPP9fvWSW4
icV3Ra5kuq/JN6EnSue91ZPKwGZfZGko7aKrGiAyKxl8ER/Pz/9a282U31rX19bcV1uFErYHDB0Q
LDLAoRtwCV5MZkRMcq3Wq8iQh34JVCrlLVTa67cw5p03aRExtd4wgtnO0yn3ZkLIkxODxTdOieOA
ZyYEV3RsjRkTW8Zk2p5paVfkpzJVatG50iRM+z+7zzqP8ibiMSWdHigMM2AzKM9tPA5DiRDeuhXu
zuP2UzhglmwHk43MUvOZSgQ0GsVTplYxrKBFVJ5NVufn2O+ToxshKhCam2z4fbXz+jDl3U8VuNmh
0dvNmQBH4uR63A6jOLzezJRK76ixlQjRcW9sIBjW2iKRfS21VImfjTSersSaZrT0gBRCNKBDILTN
l4wdNI4x2MDedUuw95cY5VejQZ1aTmBCuPfiaqS1UdibRQshWkk56rUfFOMjW6FIHaX1mgyw6w0Y
i/8fS6lpFDXj99oy02cb3WAkp+aef1cb6jjf/wNnFdHpZV+kBAHOkRkCUwENtQHYS4tng1VRqrLg
BrFcLFQNDpEQneFqOVQYyaUhUZ8SsK6ri++oK32dacLeLIZHSVvOXwt0dcH3ZJHJMELTOssHaGm3
ysPmEEqrxua0uFs5Rr168m/NEp6aBAaSjFmXyROMjXnZOqtFqnBPiPRZPPi16G6bWJMa3OF5dpQi
```

E0STq/fAgmcwIFAvXltfkmhtvaQ1Vsrh7xhtKSURiVZrSWocXTRLum2lsV2pH3oa1rxl1Y1vsSIH
rabd0OBnLwcOHolOA4fXzDrBetAsrnYjNe1MbNBF1aubcVJVqGfQGMqlLRlQYaj944wFWV2LZFpp
LVREbVojAJHVAWrSnNWyrQuHpdGZz1obLoUtrpuH5FXaPuivMu2R8ovPRhMy6zbK2QZqzKdYbbd1
y4tFntbGura6+rR5NMgedxw9flyrO1U9l0PRlGFLehgcSB7Mk+97nXHGnCTzfpI6JLFOURnmohPV
ZPQnGBveBNjqiaawRRGND6oWuMGlnNrKoM9v1WR/TpW0MzN9+nWqLvpQIrxheeCk4ivN3X9gaZKW
UrBJzu3rJpE5UHoURtZyoEfi+zxBtF7aZxhaI8FiTWNcRvK8mX4mqm5caoX1myfmgAvZvuOk2ObS
8artmkjjjVaV71FkKeq8MZidxsXBHFBxkCFG2kxMIIkHc0mLOlzNNnVHKHDx0xNQIFYWFFBiOqxf
LjORp7+MkfxcRoGiAqDGg/Mnme1IMp4yWmFfDUCmjOqltTUdpnw/PuiD4YXFxy0wdqqltGmtOcKR
JdzwGBFFbxEyOEUfzaYDU7TjHF4GlHYYOvjcYDNbe83NXN//W9vNlQzD92i9dnU3Wh2+eUGiqJJK
GyBNRAq8+TDU01BNws6puWHKnjIO0iWjhwiCtzTiKsejiY5XcIF+ZdO5XR8NWkUlGA0NUsY5m3Hj
daPOlhS1KJD63rAyZVcOzv59/9mmytGWzws+SdikT0lN2MquOo8rIx6Kndjjf/Ak7w5fn7fXacY4
bqqeznInaCovEBvdoDoF+hj21/3LtdRZJ2mvzlK9KdJ2kGBE/abefaURNlEHDtM6aL6qO0Yq09Tp
qGyhL4GwPr6xa91cHZugvwLioiJge0/W1WlHJ7bpzFcharSdzcnhVqbDPZtVGcHZsVGG3bqWAfxg
yF5ZZxES2Nb+0yJS3m7qyLicSTBt6pY26UmMXRayvQS+zelavKc62PVXOWseDZ5A2Ldc16+XDpfl
NP/1M2NCXLbLef889rw43eCd5lwZA963T7Z5VVWEg9G5OW+eAj5WmCxuNNTwURbtUXGAbgQktJlO
NN/GUzWhuhtpBQwWNGcwisaAKUaaxHSlVeIhJqlxVHChVWzd9HWreVjPwMxgzOaD28M4//wUGgse
ByHRlI9eYVqxtpCvalVmALEproJUhqu5sNVOobJGQuB2je20opqoworkibxqSzwsKh910/Q1ydHV
i9aaRyTMtCRPOdjNLqcuompCLS+jIzD4XJeJonZpZLZ9ZZqCb1XU7/MbaOmtK3pvEI1pm+N52419
ihHbcipz3V8cwVPfIKhMNV9wuI2muCoj4xWXYuFfiG+ariBteXJzn992RqnRLEkkCyrBqIvEisWX
PsbML6VsfXyFC2ny2K/B27NjNRz2sYw1nHJzJcGHT/N89q411uL7x5onJZygeXVXW2dGtratI3R5
tAUE0C7XpXF9hZtPJNKY0vX5O+CB3CdVaP0pn+VH04uF7vSyzNDA15QjdAePgee8ItvqwhsAJaV1
h4IzEDo8P/2dmbmkSGSoLc4lcdNFFjKpqKBVFtXHAi1CLa5PMZFjY5NFuLKUvS05+ZfOu01UbqNF
6m13IdrpLOl/Jzm7dwgRxLdgoEO7R4UPL8s9fuJXaSlx7aF5fkm4K2qpCJFxL0RDuyCzFWUhmvLk
1j1IrQdJOrYmoh+sbjYaysbFhavW6PYGJbKo4XAcW4PX6wsI0qmHBedBxpypwR8dj162Eeq3FQWl
D4wuNsL60ZRHqUTSa+tancKplVHZItCZltEe54cqEuK+TJJIxSkAhHULYM3J5J909dmTThS7q5EK
nFr25/vJVV//y8FlrrL/327+1NNzYeC29F4w0aaOLjtfUYGHMmRGcY+qGNRF5L/Id8cBAH5uMJ7b
qwbS6bPaZpG0t4ewZUuVqRTAlvh+Hc1KteWuTYODms1QbTFU9MZxwGmU8faFyfD02YapYy51y4mY
57cyzIjE889YVZDuL0tTbW9QPjJTKCEURqW6LO87STCNT8ndC8nul4Ajvdta6xT2SwwHFAxc0pMB
If5fX1oiS6q5SrIjMMgdgxifcEs7ZlsEXNctx6VBXKEzdYDvMnLSjewVD1WhlIw8DnKTGvDmWmdR
N54m1Lmtx8QHry3H0xJHeaMzOMBxRZfNgyH7Ohe8BBKGcVjXwsjBObeIj9so0nMuIX2HXtpe3kMh
Em5fMoKxT3DVCb1J3v9Z11hGBpYSMop9nf/1M2dGXxVhvu/6m105H43ndp1p36gGkV6VG0loPPY2
30Fii+1hUbQBdtqanuRg9YVPG/ZmNM2SQImepedmKXlpb7TaSqKpIicvSVopUu7vtuj6xJVGNqUG
ktQuuNdb9tcgCnHKiRovE6bzzzpV+iAC1qohFIEbONRwGa7OU1ZrPwTGVNESAGQbbTyZhQZk1MO6
c2GGFs2lUZblQ2kpH0ZSWQujBGRxdXpMvhBQYnwt1qX/uRIHlOt7CtffyCB9EA6jguovGUlwbwyA
6RQydEc09WgFMkCoXWA3VYI98kLrdD/NGdIglk47sSiuu3e23uuayLwHDd6dFGRd14bmB5+qia4i
W6PKqCqbS40rb4wpHyiylUlBz04xjNzjCHVOdMfcMzMAGVNnMzFa75AlfUtVl6Tgbqbaz2Ye+c95
bCff56me1sa6urbwGzaLKdHuYkjk7A2ss6iJKhJAiFZdNqcHyQeKmis7WOP+ydj+6vb5BG4cE+1I
0tVVD5JawWiF93BJWK+A/SR3P2rgIX8SoeioMY+vrFkkDSTQzNLLK2DvY+Iax0O7dL1TZj6LseF8
9HJzfhs+ZuxQp9W4ratROJ6OQtk7lhK3bI7uFURBSJaVG+8brei6/UgCFmEfHQZ4WI7nx5ptkZTj
VDm2HTJ64flPnunlgxQyTKWqbwrbi4XVnc3obWTJrSm2sb6jEiTXoEpNQwSspoKEi6uhLcYkjCwY
ghFojtRpnzfqdpIhSMxPqQVNrWCz1TaJDnowrOgWimeoyquLj+zk3zg1i9PQg4Wky6mXIy/g8Ff/
GBby9gxwXb9XVDILN56pp5O1SyWJtcfOm5Wetsb7+m181Hw7lZUWlCHrdw6JemQyXpLB6wJEijao
aQgupsd0FMDinwXGRb9mheSXWfcxYG2EjWUtVUk0iDq8xbqgrG6JXim6hKr/WhFExQLcWU4+DMTT
XEU1laW3lLqYz1sH7bh705Oxznx+hZ013g1plbCKk7gsLBm4Q28421U5w+D0IxqpjIGF0TNBA8ih
kKCnCAd3pGYRwCn2lEOH+9dqbVFa0k55FppA3Q+z1WdeL3pd2x2hqCC3NJW3FlDo8CncnEYGEear
SE7O4ZZ1FS6YCa7moEkSXNugqCo2hpYgNr3RVxVw2b4uegsFMR4z8UAqB8aQGu1TCem7Iq9K2oYc
/x3qvwVl1wL2Pitj1TyUKwXVHNN11HU1Hff9YHHDRoSkeu0BZu+nYTpcUvbn5T3zgV369eaZs6Rk
f3YRD9JLxdHpiZYglLlXyFjpS2TqLqakYP7St3nU4bxDVeMNG0OmJcDc1WHSQNy8tmSSMz9yU25n
Vs3hgB35nks4M610QjV1ei8xlrUJrrD8zR+/XM5KUSObPOmWHbN95Slys2+Z4tl4xHD7UlN1Glmv
5YjaKgzLYchLHVZVDDxvl57ZZiGYRtYP+bfRQjQu55ebb5clDR57O1+ymoB0YHdUX4YCaVsIl656
Gur0n8aCawePwApureZEpDsJgjdeZd+pBb8B1GzKZybQ4aT/xlSTWOBqNlduKLjCj4lw/rm7kjLe
eksOKOlaYbS3I1q1bYzc4zlKbJkqNOp1tBj6ZttDCWSZNZ4dsls78/LE4cXbOheyglZCa/8IzaUv
PeGQNH+ilo+G64ih9ZiXcLKQymDyUJpAlehC8kcFqNbHRhzRxxYPOvBqrZxNCzxBRVUV+SjU8NYq
yjRTSCj3r9xJd7VjrGkTO1jId0GdikT+VqGbIzLFu4szFpUuQEvbCv7v0xRcQOVUW0zaNTC2NR/Y
AgMNk3FLJkAKjy102KhIW3m+weYsCWhFVbZ9qTa5vwUdFGVWp2ntLLHXBFPiz//Q5JaJ3DCuVVpa
ErB8HInQU/TG7LKYq29rdkp4kx+yFSbQOxXOcjsZx1ylwvWT/WMc9yyp9bq5an6RR38n35tBh+za
3NDEoohoF+KXTsogQTaCZMmvy+fLkoqhlvj/PmSpq16fhrm0c2BImnD1TKa3QzhLFwhsHnHQICqj

```
p62bGyXQZWsc6HzyiuSxf/pXf/v3VZ9KWnvHIOlgcvBwEclfar0wriAZpagK1EZJpxEiVUati5UX
Tij+jEzqXqYRDItroce0gAL8HIQViQ8EN0VZGRHrXjMe0jrCPGkJOitGBI33coIioqXTjaNJ9YJ1
VjYhem5bmZfeFF7cdY2t8JV4oVvoZUJv65jlQ8HDQi/A1Yl8XNqmoeNNKu+iSZXjunmYdBoTALFr
7XEowv+vOPfL4U4dmUjTppuKouTrspC4wwbZTxLFU4zW9dLqiPq8Ml+UzT/2stfQa2xGUKr6acAm
TA4V0qHNiXWBNOtBF1vFW0tQ2d58W47aGBEAmbYXhcJ94pgYLsrBpiVzXbg06A2JIZnsQ/llYYxs
zqKraTn53juqqD12BuFj/J8c4NXzwNC/y3txPPdO29IxH1oWFpUsG82uytjamxAg4mdT59UXBidZ
vmvL5HFL5qI4ASm38AQ8LVk5tC5Y4KA8PbMkWCs+kKKl2kbM0t46uRFnM2udsMUrWLTGZkF0dCSU
dhoh3MrstfLF9Ouc/7yzZvuPkjgq+1UL+poancGHphDVXqFWHazIBnjcY7pZt5+i272DASYlVldo
QmawfDhF7nhrKXnvWZCalbDRhdeM//MO12kVPtcvbYp+bp/MvxRylobo407VGk3ZKJapUJUk6M6o
A+yUyfxpfwKSZEI4Sxk2MgqI0MbRbU2OySIQs6lLjJpLnCwqx615bLEIuo1xskmmvgM4yg5EuxVT
XWVXdDW9eH9e+Xd3mz00iMxEy/lmpllTemct2Db+LVOoafPdrWJsyl/cO9frZJ59pW3rGI2sxHJ+
6NL+gW617yqgZDSdx1QMXGosq0pX17LDZxZ81vVMldBdvrunQejI/N5gRMosDb8fbJqoL5J62tu6
y73D1jeMu5xFhd/Z8Shsp+GihSzqGM0En+NJLLw3GNODD99EA436tIm+sTdm8bUcCADZg7WIx1Ro
VeGDUq9PJmuRz87K+fFD2PXCr9OcXZeXAYzoPrV0mVLuk6ALfe89dcs/DD+nnsU1nGjG149x0D5k
ZqkeM+FlrbQ410gqlld19OqnOwOMKxk5nlwZiCCexDrcDDdqmXdvMkQ7/s7ZGWWO8XPxbOATfKse
F2uhIh3MTnD43B/prA4c8sVqfdMuqisqCXSkW/z3br8gMO14/zQhnzl89Kz/abRy2GQoRXXf+pw9
/dPqsNtY3ftfr+sPxSLVz8gGXTUGVHg+fLfJSWSVV1RyTwhhYn9uym2gc2JIdCctZxnoFy5J9hQF
urjat9O95kn6JZcLyJeuOmcZW22OJf854ZG+thAh5wuK8nHfB+cZnbNUCGzuknlZqaLbInWmTpbG
xBA7lwrYd4aCuS96b1w7o3ILMbT6BQP7x2jKWALWHOxE7pBP5zD9+Xue3EQndNkrIlqlq05dqItj
fjTb19R+pGiScpu/q8ZlnZpKdvM7O/tp4X7P2nsJYdE9MHjV4u4bAn8nTCDaITZ3URcoKjmx6Hcq
Z7rHJ41b4wlg2cc2HY3JdID7T63C9JOecdq8hmUFh5Xwuvmy5mzk5Eq6rGdxtLplDWAle5K1fjhH
o/xJgAHavyI3ijlo8AAAAAElFTkSuQmCC',
        height: 150, width: 180, top: 290, left: 50}],
    }
  },
  methods: {
 created: function () {
      var spreadsheet = this.$refs.spreadsheet;
            spreadsheet.merge('B2:D2');
            spreadsheet.merge('B11:D11');
            spreadsheet.cellFormat({ fontWeight: 'bold', verticalAlign:
'middle', backgroundColor: '#1167b1', color: '#ffffff' }, 'B2');
            spreadsheet.cellFormat({ fontWeight: 'bold', verticalAlign:
'middle', backgroundColor: '#1167b1', color: '#ffffff' }, 'B11');
            spreadsheet.cellFormat({ fontWeight: 'bold' }, 'C3:C9');
            spreadsheet.cellFormat({ fontWeight: 'bold' }, 'C12:C18');
            spreadsheet.setBorder({ border: '1px solid #1167b1' }, 'B2:D9',
'Outer');
            spreadsheet.setBorder({ border: '1px solid #1167b1' },
'B11:D18', 'Outer');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/spreadsheet/link-cs1" %}

Limitations of Image

The following features have some limitations in Image:

- Corner resizing option in the image element.
- Copy and paste the external image.

Chart

A chart is a graphical representation of data, that organizes and represents a set of numerical or qualitative data. It mostly displays the selected range of data in terms of x-axis and y-axis. You can use the allowChart property to enable or disable the chart functionality.

* The default value for the allowChart property is true.

Types of chart

The following types of charts are available in the Spreadsheet.

* Column Chart

* Bar Chart

* Area Chart

* Line Chart

* Pie Chart

* Scatter Chart

Insert Chart

You can insert the chart by using one of the following ways,

- Select the chart icon in the Ribbon toolbar under the Insert Tab.
- Use the insertChart() method programmatically.

The available parameter in the insertChart() method is,

| Parameter | Type | Description |

|-----|------|----|

| chart | ChartModel | Specifies the options to insert a chart in the spreadsheet. |

The available arguments in the ChartModel are:

- type: Specifies the type of chart.
- theme: Specifies the theme of a chart.
- isSeriesInRows: Specifies to switch the row or a column.
- range: Specifies the selected range or specified range.
- id: Specifies the chart element id.
- markerSettings: Specifies the marker settings. The marker is used to provide information about the data points in the series and is currently only applicable to the line chart.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :created="created" :height=515>
        <e-sheets>
            <e-sheet :name="BookSales">
                <e-rows>
                    <e-row :height="height" :cells="cells1"></e-row>
                    <e-row :height="height" :cells="cells2"></e-row>
                </e-rows>
                <e-ranges>
                    <e-range :dataSource="dataSource" startCell="A3" ></e-
range>
                </e-ranges>
                <e-columns>
                    <e-column :width=110></e-column>
                    <e-column :width=100></e-column>
                    <e-column :width=100></e-column>
                    <e-column :width=100></e-column>
                    <e-column :width=100></e-column>
                    <e-column :width=100></e-column>
                </e-columns>
            </e-sheet>
        </e-sheets>
    </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin, getFormatFromType } from "@syncfusion/ej2-vue-
spreadsheet";
import { chartData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      height: 30,
      BookSales:"Book Sales",
      dataSource: chartData,
      cells1: [
        { value: 'Book Sales 2016-2020', style: { backgroundColor:
'#357cd2', color: '#fff',
            fontWeight: 'bold', textAlign: 'center', verticalAlign: 'middle'
}}
        ]
    }
  },
  methods: {
  created: function () {
      var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.cellFormat({ backgroundColor: '#357cd2', color: '#fff',
fontWeight: 'bold', textAlign: 'center' }, 'A3:F3');
      spreadsheet.numberFormat(getFormatFromType('Currency'), 'B4:F8');
      spreadsheet.merge('A1:F1');
      //Render Column chart
      spreadsheet.insertChart([{type: 'Column', theme: 'Bootstrap5Dark',
range: 'A3:B6', id: 'column-chart' }]);
      //Render Line chart with Marker
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      spreadsheet.insertChart([{type: 'Line', range: 'A3:B6',
markerSettings: {visible: true, shape: 'Circle', isFilled: false, size: 10,
border: {width: 2, color: '#3cb371'}}, id: 'line-chart'}]);
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/chart-cs2" %}

Delete Chart

- If you want to delete the chart, just select the chart, and then press the Delete key.
- Use the deleteChart() method programmatically.

The available parameter in the deleteChart() method is,

| Parameter | Type | Description |

|-----|------|----|

| id | string | Specifies the id of the chart element to be deleted. |

Chart Customization

Chart feature allows you to view and insert a chart in a spreadsheet, and you can change the height and width of the chart by resizing and moving it to another position.

- You can change the height and width of the chart by resizing.
- You can change the position of the chart by drag and drop.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :created="created" :height=515>
        <e-sheets>
            <e-sheet :name="BookSales">
                <e-rows>
                    <e-row :height="height" :cells="cells1"></e-row>
                    <e-row :height="height" :cells="cells2"></e-row>
                </e-rows>
                <e-ranges>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <e-range :dataSource="dataSource" startCell="A3" ></e-
range>
                </e-ranges>
                <e-columns>
                    <e-column :width=110></e-column>
                    <e-column :width=100></e-column>
                    <e-column :width=100></e-column>
                    <e-column :width=100></e-column>
                    <e-column :width=100></e-column>
                    <e-column :width=100></e-column>
                </e-columns>
            </e-sheet>
        </e-sheets>
    </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin, getFormatFromType } from "@syncfusion/ej2-vue-
spreadsheet";
import { chartData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      height: 30,
      BookSales:"Book Sales",
      dataSource: chartData,
      cells1: [
        { value: 'Book Sales 2016-2020', style: { backgroundColor:
'#357cd2', color: '#fff',
            fontWeight: 'bold', textAlign: 'center', verticalAlign: 'middle'
}}
        ],
    cells2: [
        { index: 7, chart: [{ type: 'Column', range: 'A3:F8' }] }
        ]
    }
  },
  methods: {
  created: function () {
      var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.cellFormat({ backgroundColor: '#357cd2', color: '#fff',
fontWeight: 'bold', textAlign: 'center' }, 'A3:F3');
      spreadsheet.numberFormat(getFormatFromType('Currency'), 'B4:F8');
      spreadsheet.merge('A1:F1');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/chart-cs1" %}

Customization of line chart markers

Using the actionBegin event, you can change the shape, size, fill color, and border of the line chart marker. In the following example, you can see the modified marker appearance, such as shape and size, while creating the line chart with UI interaction.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :created="created"
:actionBegin="actionBegin" :height=515>
        <e-sheets>
            <e-sheet :name="BookSales">
                <e-rows>
                    <e-row :height="height" :cells="cells1"></e-row>
                    <e-row :height="height" :cells="cells2"></e-row>
                </e-rows>
                <e-ranges>
                    <e-range :dataSource="dataSource" startCell="A3" ></e-
range>
                </e-ranges>
                <e-columns>
                    <e-column :width=110></e-column>
                    <e-column :width=100></e-column>
                    <e-column :width=100></e-column>
                    <e-column :width=100></e-column>
                    <e-column :width=100></e-column>
                    <e-column :width=100></e-column>
                </e-columns>
            </e-sheet>
        </e-sheets>
    </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin, getFormatFromType } from "@syncfusion/ej2-vue-
spreadsheet";
import { chartData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      height: 30,
      BookSales:"Book Sales",
      dataSource: chartData,
      cells1: [
        { value: 'Book Sales 2016-2020', style: { backgroundColor:
'#357cd2', color: '#fff',
```

```
            fontWeight: 'bold', textAlign: 'center', verticalAlign: 'middle'
}}
        ]
    }
  },
  methods: {
        created: function () {
            var spreadsheet = this.$refs.spreadsheet;
            spreadsheet.cellFormat({ backgroundColor: '#357cd2', color:
'#fff', fontWeight: 'bold', textAlign: 'center' }, 'A3:F3');
            spreadsheet.numberFormat(getFormatFromType('Currency'),
'B4:F8');
            spreadsheet.merge('A1:F1');
        },
        actionBegin: function (args) {
            if (args.action === 'beforeInsertChart' &&
args.args.eventArgs.type.includes('Line')) {
                args.args.eventArgs.markerSettings.shape = 'Triangle';
                args.args.eventArgs.markerSettings.isFilled = false;
                args.args.eventArgs.markerSettings.size = 10;
            }
        }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/chart-cs3" %}

Limitations of Chart

The following features have some limitations in the Chart:

- Insert row/delete row between the chart data source will not reflect the chart.
- Copy/paste into the chart data source will not reflect the chart.
- Corner resizing option in chart element.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

Copyright © 2001 -2024 Syncfusion Inc.

See Also

- Formatting
- Rows and columns
- Hyperlink
- Sorting

Rows and columns in Vue Spreadsheet component

Spreadsheet is a tabular format consisting of rows and columns. The intersection point of rows and columns are called as cells. The list of operations that you can perform in rows and columns are,

- Insert
- Delete
- Show and Hide

Insert

You can insert rows or columns anywhere in a spreadsheet. Use the allowInsert property to enable or disable the insert option in Spreadsheet.

Row

The rows can be inserted in the following ways,

- Using insertRow method, you can insert the rows once the component is loaded.
- Using context menu, insert the empty rows in the desired position.

The following code example shows the options for inserting rows in the spreadsheet.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false"
                :showSheetTabs="false" :showRibbon="false">
                <e-sheets>
                  <e-sheet>
                    <e-ranges>
                      <e-range :dataSource="dataSource" startCell="B1"></e-
range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=20></e-column>
                      <e-column :width=90></e-column>
                      <e-column :width=220></e-column>
                      <e-column :width=90></e-column>
                      <e-column :width=140></e-column>
                      <e-column :width=90></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                    </e-columns>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data  } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
       // Rows model that is going to insert dynamically
       rowsModel: [
           {
               index: 9, // Need to specify the index for the first row
collection, the specified rows will be inserted in this index.
               cells: [{ value: '' }, { value: '8' }, { value: 'Northwoods
Cranberry Sauce' }, { value: '3' }, { value: '12 - 12 oz jars' },
                   { value: '40.00' }, { value: '6' }, { value: 'false' }]
           },
           {
               cells: [{ value: '' }, { value: '9' }, { value: 'Mishi Kobe
Niku' }, { value: '4' }, { value: '18 - 500 g pkgs.' },
                   { value: '97.00' }, { value: '29' }, { value: 'true' }]
           }
         ]
    }
  },
  methods: {
   created: function () {
        var spreadsheet = this.$refs.spreadsheet;
     // Applies style formatting before inserting the rows
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'B1:H1');
        // inserting a empty row at 0th index
        spreadsheet.insertRow();
        // inserting 2 rows at the 9th index with data
        spreadsheet.insertRow(this.rowsModel);
        // Applies style formatting after the rows are inserted
        spreadsheet.cellFormat({ textAlign: 'center' }, 'B3:B12');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'D3:D12');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'F3:H12');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/insert-row-cs1" %}

Column

The columns can be inserted in the following ways,

- Using insertColumn method, you can insert the columns once the component is loaded.
- Using context menu, insert the empty columns in the desired position.

The following code example shows the options for inserting columns in the spreadsheet.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false"
                 :showSheetTabs="false" :showRibbon="false">
                 <e-sheets>
                   <e-sheet>
                     <e-ranges>
                       <e-range :dataSource="dataSource" startCell="A2"></e-
range>
                     </e-ranges>
                     <e-columns>
                       <e-column :width=90></e-column>
                       <e-column :width=220></e-column>
                       <e-column :width=90></e-column>
                       <e-column :width=140></e-column>
                       <e-column :width=100></e-column>
                       <e-column :width=100></e-column>
                     </e-columns>
                   </e-sheet>
                 </e-sheets>
               </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin, getCellAddress } from "@syncfusion/ej2-vue-
spreadsheet";
import { data  } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
      // Applies style formatting before inserting the column
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A2:G2');
        // inserting a empty column at 0th index
```

```
        spreadsheet.insertColumn();
        // inserting 1 column at the 5th index with column model
        spreadsheet.insertColumn([{ index: 5, width: 90 }]);
        let rowIndex = 1;
        // Updating the 5th column data
        var cellsModel=  [{ value: 'Unit Price', style: { fontWeight:
'bold', textAlign: 'center' } }, { value: '18.00' },
        { value: '19.00' }, { value: '10.00' }, { value: '22.00' }, { value:
'21.35' }, { value: '25.00' }, { value: '30.00' },
        { value: '21.00' }, { value: '40.00' }, { value: '97.00' }]
         cellsModel.forEach((cell) => {
            spreadsheet.updateCell(cell, getCellAddress(rowIndex, 5));
rowIndex++;
        });
        // Applies style formatting after the rows are inserted
        spreadsheet.cellFormat({ textAlign: 'center' }, 'B3:B12');
       spreadsheet.cellFormat({ textAlign: 'center' }, 'D3:D12');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'F3:H12');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/insert-column-cs1" %}

Delete

Delete support provides an option for deleting the rows and columns in the spreadsheet. Use allowDelete property to enable or disable the delete option in Spreadsheet.

The rows and columns can be deleted dynamically in the following ways,

- Using delete method, you can delete the loaded rows and columns.
- Using context menu, you can delete the selected rows and columns.

The following code example shows the delete operation of rows and columns in the spreadsheet.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false"
                :allowDelete="true" :showRibbon="false">
```

```
                <e-sheets>
                  <e-sheet name="Sheet1">
                    <e-ranges>
                      <e-range :dataSource="dataSource" ></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=90></e-column>
                      <e-column :width=220></e-column>
                      <e-column :width=90></e-column>
                      <e-column :width=140></e-column>
                      <e-column :width=90></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                    </e-columns>
                  </e-sheet>
                  <e-sheet name="Sheet2">
                    <e-ranges>
                      <e-range :dataSource="dataSource" ></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=90></e-column>
                      <e-column :width=220></e-column>
                      <e-column :width=90></e-column>
                      <e-column :width=140></e-column>
                      <e-column :width=90></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                    </e-columns>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data  } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
        // deleting the rows from 8th to 10th index. To delete row, the
third argument of enum type is passed as 'Row', the last argument specifies
the sheet name or index in which the delete operation will perform. By
default,active sheet will be considered. It is applicable only for model
type Row and Column.
        spreadsheet.delete(8, 10, 'Row', 0); // startIndex, endIndex, Row,
sheet index
        // deleting the 2nd and 5th indexed columns
        spreadsheet.delete(2, 2, 'Column', 'Sheet2');
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        spreadsheet.delete(5, 5, 'Column');
        spreadsheet.delete(0, 0, "Sheet"); // delete the first sheet. sheet
index starts from 0
        // Applies style formatting after deleted the rows and columns
        spreadsheet.cellFormat({ textAlign: 'center' }, 'A2:A8');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'D2:G8');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/delete-row-cs1" %}

Limitations of insert and delete

The following features have some limitations in Insert/Delete:

- Insert row/column between the formatting applied cells.
- Insert row/column between the data validation.
- Insert row/column between the conditional formatting applied cells.
- Insert/delete row/column between the filter applied cells.

Hide and show

You can show or hide the rows and columns in the spreadsheet through property binding, method, and context menu.

Row

The rows can be hidden or shown through the following ways,

- Using `hidden` property in row, you can hide/show the rows at initial load.
- Using `hideRow` method, you can hide the rows by specifying the start and end row index, set the last argument `hide` as `false` to unhide the hidden rows.
- Right-click on the row header and select the desired option from context menu

Column

The columns can be hidden or shown through following ways,

- Using `hidden` property in columns, you can hide/show the columns at initial load.

Copyright © 2001 -2024 Syncfusion Inc.

- Using hideColumn method, you can hide the columns by specifying the start and end column index, set the last argument hide as false to unhide the hidden columns.
- Right-click on the column header and select the desired option from context menu

The following code example shows the hide/show rows and columns operation in the spreadsheet.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false"
                :showSheetTabs="false" :showRibbon="false">
                <e-sheets>
                  <e-sheet>
                    <e-ranges>
                      <e-range :dataSource="dataSource"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=150></e-column>
                      <!-- Hiding the 1st and 2nd column index through
property binding -->
                      <e-column :width=100 :hidden="true"></e-column>
                      <e-column :width=100 :hidden="true"></e-column>
                      <e-column :width=80></e-column>
                      <e-column :width=80></e-column>
                      <e-column :width=80></e-column>
                      <e-column :width=80></e-column>
                      <e-column :width=80></e-column>
                    </e-columns>
                    <e-rows>
                      <e-row index=2 :hidden="true"></e-row>
                      <e-row :hidden="true"></e-row>
                    </e-rows>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data  } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
        // Unhide the 2nd index hidden column
        spreadsheet.hideColumn(1, 1, false);
        // Unhide the 3rd index hidden row
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        spreadsheet.hideRow(3, 3, false);
        // Hiding the 6th index column
        spreadsheet.hideColumn(6);
        // Hiding the 8th and 9th index row
        spreadsheet.hideRow(8, 9);
        spreadsheet.cellFormat({ textAlign: 'center' }, 'D2:H11');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/show-hide-cs1" %}

Size

You can change the size of rows and columns in the spreadsheet by using setRowsHeight and setColumnsWidth methods.

Row

You can change the height of single or multiple rows by using the setRowsHeight method.

You can provide the following type of ranges to the method:

- Single row range: ['2:2']
- Multiple rows range: ['1:100']
- Multiple rows with discontinuous range: ['1:10', '15:25', '30:40']
- Multiple rows with different sheets: [Sheet1!1:50, 'Sheet2!1:50', 'Sheet3!1:50']

The following code example shows how to change the height for single/multiple rows in the spreadsheet.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created">
                <e-sheets>
                  <e-sheet>
                    <e-ranges>
                      <e-range :dataSource="dataSource"></e-range>
                    </e-ranges>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data  } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
       // To change height for single row
       spreadsheet.setRowsHeight(40, ['2']);
       // To change height for multiple rows
       spreadsheet.setRowsHeight(40, ['4:8', '10:12']);
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/row-height-cs1" %}

Column

You can change the width of single or multiple columns by using the setColumnsWidth method.

You can provide the following type of ranges to the method:

- Single column range: [‘F:F’]
- Multiple columns range: [‘A:F’]
- Multiple columns with discontinuous range: [‘A:C’, ‘G:I’, ‘K:M’]
- Multiple columns with different sheets: [Sheet1!A:H, ‘Sheet2!A:H’, ‘Sheet3!A:H’]

The following code example shows how to change the width for single/multiple columns in the spreadsheet.

APP.VUE

```
<template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  <ejs-spreadsheet ref="spreadsheet" :created="created">
                <e-sheets>
                  <e-sheet>
                    <e-ranges>
                      <e-range :dataSource="dataSource"></e-range>
                    </e-ranges>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data  } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
       // To change width of single column
       spreadsheet.setColumnsWidth(100, ['F']);
       // To change width of multiple columns
       spreadsheet.setColumnsWidth(120, ['A:C', 'G:I', 'K:M']);
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/column-width-cs1" %}

Changing text in column headers

Using the beforeCellRender event, you can change the text in the column headers. In that event, you can use the e-header-cell class to identify the header cell element and update its text value.

The following code example shows how to change the text in the column headers.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <ejs-spreadsheet ref="spreadsheet"
:beforeCellRender="beforeCellRender"></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
Vue.use(SpreadsheetPlugin);
export default {
  methods: {
    beforeCellRender: function (args) {
      // Condition to check whether the rendered element is header cell.
      if (
        args.colIndex >= 0 &&
        args.colIndex <= 10 &&
        args.element.classList.contains('e-header-cell')
      ) {
        var text = 'custom header ' + args.colIndex.toString();
        // Add the custom text to the innerText of the element.
        args.element.innerText = text;
      }
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/column-header-change-cs1" %}

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Hyperlink
- Sorting
- Filtering

Filter in Vue Spreadsheet component

Filtering helps you to view specific rows in the spreadsheet by hiding the other rows. You can use the allowFiltering property to enable or disable filtering functionality.

Copyright © 2001 -2024 Syncfusion Inc.

* The default value for allowFiltering property is true.

By default, the Filter module is injected internally into Spreadsheet to perform filtering.

Apply filter on UI

In the active sheet, select a range of cells to filter by value of the cell. The filtering can be done by any of the following ways:

- Select the filter item in the Ribbon toolbar.
- Right-click the sheet, select the filter item in the context menu.
- Use the applyFilter() method programmatically.
- Use Ctrl + Shift + L keyboard shortcut to apply the filter.

* Use Alt + Up/Down keyboard shortcut to open the filter dialog.

Filter by criteria

The applyFilter() method will apply the filter UI, based on the predicate and range given in the arguments.

* The beforeFilter event will be triggered before filtering the specified range.

* The filterComplete event will be triggered after the filter action is completed successfully.

The following code example shows filter functionality in the Spreadsheet control.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :allowFiltering='true'
:dataBound="dataBound">
   <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
            <e-columns>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width3"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width3"></e-column>
              <e-column :width="width1"></e-column>
              <e-column :width="width1"></e-column>
              <e-column :width="width4"></e-column>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
              <e-column :width="width5"></e-column>
            </e-columns>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { tradeData } from './data.js';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: tradeData,
      width1: 100,
      width2: 130,
      width3: 96,
      width4: 110,
      width5: 150,
    }
  },
  methods: {
  dataBound: function () {
      var spreadsheet = this.$refs.spreadsheet;
        if (spreadsheet.ej2Instances.activeSheetIndex === 0) {
            var departments = ['Sweden', 'Canada', 'UK'];
            var predicateList = []
            departments.forEach((department) => { predicateList.push({
field: 'D', predicate: 'or', operator: 'equal', value: department }); })
            spreadsheet.applyFilter(predicateList);
        }
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/filter-cs1" %}

Filter by cell value

To apply a filter for a cell value, right-click the cell and choose filter -> Filter By Selected Cell's Value option from the menu. It applies the filter based on the value of the selected cell in the current sheet.

Clear filter

After applying filter to a certain column, you may want to clear it to make all filtered rows visible again. It can be done in the following ways,

- Choose Clear option in ribbon toolbar under Filter and Sort. It clears the filters applied in the spreadsheet for all fields.
- Use the clearFilter() method programmatically, to clear the applied filters in spreadsheet for all fields.

Copyright © 2001 -2024 Syncfusion Inc.

Clear filter on a field

After filtering, you can clear/reset the filter for a field alone. It can be done in the following ways,

- Click filter icon in the column's header and then choose Clear Filter option from the filter dialog.
- You can right-click on a filtered column cell and choose Clear Filter from . option from the context menu.
- Use the clearFilter(field) method programmatically, to clear the filter in a particular column.

Reapply filter

When you want to reapply the filter after some changes happened in the rows. It can be done in the following ways,

- You can choose Reapply option in ribbon toolbar under Filter and Sort to reapply the filtered columns again.
- You can right-click on a filtered cell and choose Reapply option from the context menu. It reapplies the filters again in the Spreadsheet for all the fields.

Known error validations

The following errors have been handled for filtering,

- *Out of range validation:* When the selected range is not a used range of the active sheet, it is considered as invalid and the out of range alert with the message Select a cell or range inside the used range and try again will be displayed. No filter will be performed if the range is invalid.

Limitations

The following features have some limitations in Filter:

- Insert/delete row/column between the filter applied cells.
- Merge cells with filter.
- Copy/cut paste the filter applied cells.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Sorting
- Hyperlink
- Undo Redo

Sort in Vue Spreadsheet component

Sorting helps arranging the data to a specific order in a selected range of cells. You can use the allowSorting property to enable or disable sorting functionality.

Copyright © 2001 -2024 Syncfusion Inc.

* The default value for allowSorting property is true.

By default, the sort module is injected internally into Spreadsheet to perform sorting.

Sort by cell value

In the active Spreadsheet, select a range of cells to sort by cell value. The range sort can be done by any of the following ways:

- Select the sort item in the Ribbon toolbar and choose the ascending or descending item.
- Right-click the sheet, select the sort item in the context menu and choose the ascending/descending item.
- Use the sort() method programmatically.

The cell values can be sorted in the following orders:

- Ascending
- Descending

* Ascending is the default order for sorting.

The sort() method with empty arguments will sort the selected range by active cell's column as sort column in ascending order.

* The beforeSort event will be triggered before sorting the specified range.

* The sortComplete event will be triggered after the sort action is completed successfully.

The following code example shows Sort functionality in the Spreadsheet control.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :allowSorting='true'
:dataBound="dataBound" :beforeSort="beforeSort"
:sortComplete="sortComplete">
   <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: defaultData,
    }
  },
  methods: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    dataBound: function () {
      var spreadsheet = this.$refs.spreadsheet;
        if (spreadsheet.ej2Instances.activeSheetIndex === 0) {
            spreadsheet.sort({ containsHeader: true }, 'A1:H11');
        }
      },
    beforeSort: function (args) {
        //code here to handle sorting arguments.
    },
    sortComplete: function (args) {
        var spreadsheet = this.$refs.spreadsheet;
        spreadsheet.selectRange(args.range);
        // code here.
    },
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/sort-by-cell-cs1" %}

Data contains header

You can specify whether the selected range of cells contains header. To specify, you need to set the containsHeader property to true and pass it as sortOption arguments of the sort() method.

* If the containsHeader property is not set and active cell column's first cell value type is differed from the second cell value type, the first row data in the range are marked as column headers.

You can also enable or disable this property using beforeSort event arguments,

`ts

beforeSort: function (args) {

args.sortOptions.containsHeader = true;

}

`

In the custom sort dialog, the Data contains header checkbox is checked on load. Thus, the default value for containsHeader is true in custom sort dialog.

Copyright © 2001 -2024 Syncfusion Inc.

Case sensitive sort

The default sort functionality of Spreadsheet is a case insensitive sorting. When you want to perform sorting with case sensitive, you need to set the caseSensitive property to true and pass it as sortOption arguments of the sort() method.

Case sensitive sorting is applicable only for cells with alphabets. In ascending order sorting with case sensitive enabled, the cells with lower case text will be placed above the cells with upper case text.

* The default value for the caseSensitive property is false.

You can also enable or disable this property using beforeSort event arguments,

`ts

beforeSort: function (args) {

args.sortOptions.caseSensitive = true;

}

`

In the custom sort dialog, the Case sensitive checkbox is unchecked on load as the default value is false.

Sort multiple columns

When you want to perform sorting on multiple columns, it can be done by any of the following ways:

- Select the Custom sort… menu item from the Ribbon toolbar item or context menu item.
- Use the sort() method programmatically by providing sort criteria.

* The current sorting functionality supports sorting based on cell values only.

Custom sort dialog

The custom sort dialog helps sorting multiple columns in the selected range by utilizing the rich UI. This dialog will be appeared while choosing the Custom sort… from the Ribbon item or context menu item. By default, sort criteria with the first column name from the selected range will be appeared in the dialog on initial load and it cannot be removed.

You can add multiple criteria using the Add Column button at the bottom of the dialog. Thus, multiple columns can be specified with different sort order. The newly added sort criteria items can be removed using the delete icons at the end of each items.

You can refer to the Data contains header topic to learn more about Data contains header checkbox. To learn more about Case sensitive checkbox, you can refer to Case sensitive sort topic.

Passing sort criteria manually

The multi-column sorting can also be performed manually by passing sort options to the sort() method programmatically. The sortOption have the following arguments:

- sortDescriptors – Sort criteria collection that holds the collection of field name, sort order, and sortComparer.
- containsHeader – Boolean argument that specifies whether the range has headers in it.
- caseSensitive – Boolean argument that specifies whether the range needs to consider case.

Copyright © 2001 -2024 Syncfusion Inc.

* All the arguments are optional.

* When a sortDescriptor is specified without field, the field of the first sortDescriptor from the collection will be assigned from active cell's column name and others will be ignored. Hence, it will act as single column sorting.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :allowSorting='true'
:dataBound="dataBound" :sortComplete="sortComplete">
   <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { tradeData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: tradeData,
    }
  },
  methods: {
    dataBound: function () {
        var spreadsheet = this.$refs.spreadsheet;
        var sortDescriptors = [
        {
            field: 'F',
            order: 'Ascending'
        },
        {
            field: 'E',
            order: 'Ascending'
        },
        {
            field: 'C',
            order: 'Descending'
        }];
        if (spreadsheet.ej2Instances.activeSheetIndex === 0) {
            spreadsheet.sort({ sortDescriptors: sortDescriptors,
containsHeader: true }, 'A1:H30');
        }
    },
    sortComplete: function (args) {
        var spreadsheet = this.$refs.spreadsheet;
        spreadsheet.selectRange(args.range);
        // code here.
    },
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/passing-sort-cs1" %}

Custom sort comparer

The sortDescriptor holds the sortComparer property, which is a function and it is used to customize the sort comparer for specific sort criteria. Each sortDescriptor can be customized using the custom sort comparer function.

By customizing sort comparer, you can define the sort action as desired.

* The sortComparer is an optional property of sortDescriptor.

For custom sort comparer example, refer to the Sort a range by custom list in the how-to section.

Known error validations

The following errors have been handled for sorting,

- *Out of range validation:* When the selected range is not a used range of the active sheet, it is considered as invalid and the out of range alert with the message Select a cell or range inside the used range and try again will be displayed. No sort will be performed if the range is invalid.
- *Empty field validation:* When the sort criteria does not have a column selected (empty) in the custom sort dialog, it will become invalid, and an error message Sort criteria column should not be empty will be displayed on OK button click.
- *Duplicate field validation:* When the column names of added sort criteria are repeated more than once in the custom sort dialog, it will become invalid and an error message is mentioned more than once. Duplicate columns must be removed will be displayed on OK button click.

Limitations

- Sorting is not supported with formula contained cells.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

Copyright © 2001 -2024 Syncfusion Inc.

See Also

- Sort a range by custom list
- Hyperlink
- Filtering
- Undo Redo

Link in Vue Spreadsheet component

Hyperlink is used to navigate to web links or cell reference within the sheet or to other sheets in Spreadsheet. You can use the allowHyperlink property to enable or disable hyperlink functionality.

* The default value for `allowHyperlink` property is `true`.

Insert Link

You can insert a hyperlink in a worksheet cell for quick access to related information.

User Interface:

In the active spreadsheet, click the cell where you want to create a hyperlink. Insert hyperlink can be done by any of the following ways:

- Select the INSERT tab in the Ribbon toolbar and choose the `Link` item.
- Right-click the cell and then click Hyperlink item in the context menu.
- Use `Ctrl + K` keyboard shortcut to apply the hyperlink.
- Use the addHyperlink method programmatically.

Edit Hyperlink

You can change an existing hyperlink in your workbook by changing its destination or the text that is used to represent it.

User Interface:

In the active spreadsheet, Select the cell that contains the hyperlink that you want to change. Editing the hyperlink while opening the dialog can be done by any one of the following ways:

- Select the INSERT tab in the Ribbon toolbar and choose the `Link` item.
- Right-click the cell and then click Edit Hyperlink item in the context menu, or you can press Ctrl+K.

In the Edit Link dialog box, make the changes that you want, and click UPDATE.

Remove Hyperlink

Performing this operation remove a single hyperlink without losing the display text.

User Interface:

In the active spreadsheet, click the cell where you want to remove a hyperlink. remove hyperlink can be done by any of the following ways:

- Right-click the cell and then click Remove Hyperlink item in the context menu.
- Use the removeHyperlink() method programmatically.

Copyright © 2001 -2024 Syncfusion Inc.

How to change target attribute

There is an event named beforeHyperlinkClick which triggers only on clicking hyperlink. You can customize where to open the hyperlink by using the target property in the arguments of that event.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet"
:beforeHyperlinkClick="beforeHyperlinkClick">
   <e-sheets>
          <e-sheet name="PriceDetails">
             <e-rows>
                    <e-row>
                           <e-cells>
                               <e-cell value="Item Name"></e-cell>
                               <e-cell value="Stock Detail"></e-cell>
                               <e-cell value="Website"></e-cell>
                           </e-cells>
                       </e-row>
                       <e-row>
                           <e-cells>
                               <e-cell value="Casual Shoes"></e-cell>
                               <e-cell value="OUT OF STOCK"></e-cell>
                               <e-cell value="Amazon"
hyperlink="https://www.amazon.com/"></e-cell>
                           </e-cells>
                       </e-row>
                       <e-row>
                           <e-cells>
                               <e-cell value="Sports Shoes"></e-cell>
                               <e-cell value="IN STOCK"
hyperlink="Stock!A2:B2"></e-cell>
                               <e-cell value="Overstack"
hyperlink="https://www.overstock.com/"></e-cell>
                           </e-cells>
                       </e-row>
                       <e-row>
                           <e-cells>
                               <e-cell value="Formal Shoes"></e-cell>
                               <e-cell value="IN STOCK"
hyperlink="Stock!A3:B3"></e-cell>
                               <e-cell value="AliExpress"
hyperlink="https://www.aliexpress.com/"></e-cell>
                           </e-cells>
                       </e-row>
                       <e-row>
                           <e-cells>
                               <e-cell value="Sandals & Floaters"></e-cell>
                               <e-cell value="OUT OF STOCK"></e-cell>
                               <e-cell value="AliBaba"
hyperlink="https://www.aliBaba.com/"></e-cell>
                           </e-cells>
                       </e-row>
                       <e-row>
                           <e-cells>
```

```
                                <e-cell value="Flip-Flops & Slippers"></e-
cell>
                                <e-cell value="IN STOCK"
hyperlink="Stock!A4:B4"></e-cell>
                                <e-cell value="Taobao"
hyperlink="https://www.taobao.com/"></e-cell>
                            </e-cells>
                        </e-row>
                    </e-rows>
            <e-columns>
              <e-column :width="width1"></e-column>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
            </e-columns>
          </e-sheet>
          <e-sheet name="Stock">
             <e-rows>
                    <e-row>
                            <e-cells>
                                <e-cell value="Item Name"></e-cell>
                                <e-cell value="Available Count"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row>
                            <e-cells>
                                <e-cell value="Sports Shoes"></e-cell>
                                <e-cell value="30"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row>
                            <e-cells>
                                <e-cell value="Formal Shoes"></e-cell>
                                <e-cell value="25"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row>
                            <e-cells>
                                <e-cell value="Flip-Flops & Slippers"></e-
cell>
                                <e-cell value="40"></e-cell>
                            </e-cells>
                        </e-row>
                        <e-row>
                            <e-cells>
                                <e-cell value="Running Shoes"></e-cell>
                                <e-cell value="25"></e-cell>
                            </e-cells>
                        </e-row>
                    </e-rows>
            <e-columns>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
            </e-columns>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      width1: 160,
      width2: 120,
    }
  },
  methods: {
  beforeHyperlinkClick: function (args) {
      args.target = '_self'; //change target attribute
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/link-cs2" %}

Limitations

- Inserting hyperlink not supported for multiple ranges.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Sorting
- Filtering
- Undo Redo

Selection in Vue Spreadsheet component

Selection provides interactive support to highlight the cell, row, or column that you select. Selection can be done through Mouse, Touch, or Keyboard interaction. To enable selection, set `mode` as `Single` | `Multiple` in selectionSettings. If you set `mode` to `None`, it disables the UI selection.

* The default value for `mode` in `selectionSettings` is `Multiple`.

Copyright © 2001 -2024 Syncfusion Inc.

You have the following options in Selection,

- Cell selection
- Row selection
- Column selection

Cell selection

Cell selection is used to select a single or multiple cells. It can be performed using the selectRange method.

User Interface:

- Click on a cell to select it (or) use the arrow keys to navigate to it and select it.
- To select a range, select a cell, then use the left mouse button to select and drag over to other cells (or) use the Shift + arrow keys to select the range.
- To select non-adjacent cells and cell ranges, hold Ctrl and select the cells.

You can quickly locate and select specific cells or ranges by entering their names or cell references in the Name box, which is located to the left of the formula bar, and also select named or unnamed cells or ranges by using the Go To (Ctrl+G) command.

Row selection

Row selection is used to select a single or multiple rows.

User Interface:

You can perform row selection in any of the following ways,

- By clicking the row header.
- To select multiple rows, select a row header with the left mouse button and drag over to other row headers (or) use the Shift + arrow keys to select multiple rows.
- To select non-adjacent rows, hold Ctrl and select the row header.
- You can also use the selectRange method for row selection.

The following sample shows the row selection in the spreadsheet, here selecting the 5th row using the selectRange method.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created"
:selectionSettings="selectionSettings">
                <e-sheets>
                  <e-sheet name="Price Details">
                    <e-ranges>
                      <e-range :dataSource="dataSource1"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    </e-columns>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin, getRangeAddress } from "@syncfusion/ej2-vue-
spreadsheet";
import { priceData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
        selectionSettings: { mode: 'Multiple' },
      dataSource1: priceData
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
    var colCount = spreadsheet.ej2Instances.getActiveSheet().colCount;
        spreadsheet.selectRange(getRangeAddress([4, 0, 4, colCount]));
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/selection-cs1" %}

Column selection

Column selection is used to select a single or multiple columns.

User Interface:

You can perform column selection in any of the following ways,

- By clicking the column header.
- To select multiple columns, select a column header with the left mouse button and drag over to other column headers (or) use the Shift + arrow keys to select the multiple columns.

Copyright © 2001 -2024 Syncfusion Inc.

- To select non-adjacent columns, hold Ctrl and select the column header.
- You can also use the selectRange method for row selection.

The following sample shows the column selection in the spreadsheet, here selecting the 3rd column using the selectRange method.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created"
:selectionSettings="selectionSettings">
                <e-sheets>
                  <e-sheet name="Price Details">
                    <e-ranges>
                      <e-range :dataSource="dataSource1"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                    </e-columns>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin, getRangeAddress } from "@syncfusion/ej2-vue-
spreadsheet";
import { priceData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
        selectionSettings: { mode: 'Multiple' },
      dataSource1: priceData
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
    var rowCount = spreadsheet.ej2Instances.getActiveSheet().rowCount;
        spreadsheet.selectRange(getRangeAddress([0, 2, rowCount, 2]));
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/selection-cs2" %}

How to remove selection in the spreadsheet

The following sample shows, how to remove the selection in the spreadsheet. Here changing the mode as None in selectionSettings to disable's the UI selection.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created" :cellEdit="cellEdit"
:selectionSettings="selectionSettings">
                <e-sheets>
                  <e-sheet name="Price Details">
                    <e-ranges>
                      <e-range :dataSource="dataSource1"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                    </e-columns>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin, getRangeAddress } from "@syncfusion/ej2-vue-
spreadsheet";
import { priceData  } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
        selectionSettings: { mode: 'None' },
      dataSource1: priceData
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
      },
      cellEdit: function(args) {
        args.cancel = true;
        }
```

```
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/selection-cs3" %}

Limitations

- We have a limitation while performing the Select All(ctrl + A). You can do this only by clicking the Select All button at the top left corner.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

Clipboard in Vue Spreadsheet component

The Spreadsheet provides support for the clipboard operations (cut, copy, and paste). Clipboard operations can be enabled or disabled by setting the enableClipboard property in Spreadsheet.

By default, the enableClipboard property is true.

Cut

It is used to cut the data from selected range of cells, rows or columns in a spreadsheet and make it available in the clipboard.

User Interface:

Cut can be done in one of the following ways.

- Using Cut button in the Ribbon's HOME tab to perform cut operation.
- Using Cut option in the Context Menu.
- Using Ctrl + X | Command + X keyboard shortcut.
- Using the cut method.

Copy

It is used to copy the data from selected range of cells, rows or columns in a spreadsheet and make it available in the clipboard.

User Interface:

Copyright © 2001 -2024 Syncfusion Inc.

Copy can be done in one of the following ways.

- Using Copy button in the Ribbon's HOME tab to perform copy operation.
- Using Copy option in the Context Menu.
- Using Ctrl + C | Command + C keyboard shortcut.
- Using the copy method.

Paste

It is used to paste the clipboard data to the selected range, rows or columns. You have the following options in Paste,

- Paste Special - You can paste the values with formatting.
- Paste - You can paste only the values without formatting.

It also performs for external clipboard operation. If you perform cut and paste, clipboard data will be cleared, whereas in copy and paste the clipboard contents will be maintained. If you perform paste inside the copied range, the clipboard data will be cleared.

User Interface:

Paste can be done in one of the following ways.

- Using Paste button in the Ribbon's HOME tab to perform paste operation.
- Using Paste option in the Context Menu.
- Using Ctrl + V | Command + V keyboard shortcut.
- Using the paste method.

If you use the Keyboard shortcut key for cut (Ctrl + X) | copy (Ctrl + C) from other sources, you should use Ctrl + V shortcut while pasting into the spreadsheet.

APP.VUE

```
<template>
  <div>
    <ejs-dropdownbutton :items='items' :select='itemSelect'>Clipboard</ejs-
dropdownbutton>
    <ejs-spreadsheet ref="spreadsheet" :created="created">
      <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
             <e-columns>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
            </e-columns>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
        <div>
</template>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { SpreadsheetPlugin, getRangeIndexes } from "@syncfusion/ej2-vue-
spreadsheet";
import { DropDownButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
Vue.use(DropDownButtonPlugin);
export default {
   data: () => {
    return {
      dataSource: defaultData,
      width1: 130,
      width2: 96,
      items:[
        {
          text: "Cut"
        },
        {
          text: "Copy"
        },
        {
          text: "Paste"
        }]
    }
  },
  methods: {
  created: function () {
       var spreadsheet = this.$refs.spreadsheet;
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
      },
      itemSelect: function(args) {
      var spreadsheet = this.$refs.spreadsheet;
      if (args.item.text === 'Cut')
        spreadsheet.cut();
      if (args.item.text === 'Copy')
        spreadsheet.copy();
      if (args.item.text === 'Paste')
        spreadsheet.paste();
   }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/spreadsheet/clipboard-cs1" %}

Prevent the paste functionality

The following example shows, how to prevent the paste action in spreadsheet. In actionBegin event, you can set cancel argument as false in paste request type.

APP.VUE

```
<template>
  <div>
    <ejs-dropdownbutton :items='items' :select='itemSelect'>Clipboard</ejs-
dropdownbutton>
    <ejs-spreadsheet ref="spreadsheet" :created="created"
:actionBegin="actionBeginHandler">
      <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
             <e-columns>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
            </e-columns>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
        <div>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin, getRangeIndexes } from "@syncfusion/ej2-vue-
spreadsheet";
import { DropDownButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
Vue.use(DropDownButtonPlugin);
export default {
   data: () => {
    return {
      dataSource: defaultData,
      width1: 130,
      width2: 96,
      items:[
        {
          text: "Cut"
        },
        {
          text: "Copy"
        },
        {
          text: "Paste"
        }]
    }
  },
  methods: {
  created: function () {
       var spreadsheet = this.$refs.spreadsheet;
```

```
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
      },// Triggers before the action begins.
    actionBeginHandler: function(pasteArgs) {
      // To cancel the paste action.
        if (pasteArgs.args.eventArgs.requestType === 'paste') {
            pasteArgs.args.eventArgs.cancel = true;
        }
    },
      itemSelect: function(args) {
      var spreadsheet = this.$refs.spreadsheet;
      if (args.item.text === 'Cut')
        spreadsheet.cut();
      if (args.item.text === 'Copy')
        spreadsheet.copy();
      if (args.item.text === 'Paste')
        spreadsheet.paste();
   }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/clipboard-cs2" %}

Limitations

- External clipboard is not fully supported while copying data from another source and pasting into a spreadsheet, it only works with basic supports (Values, Number, cell, and Text formatting).
- If you copy =SUM(A2,B2) and paste, the formula reference will change depending on the pasted cell address but we don't have support for nested formula(formula reference will be same).
- Clipboard is not supported with conditional formatting (values only pasting).
- We have limitation while copying the whole sheet data and pasting it into another sheet.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

Copyright © 2001 -2024 Syncfusion Inc.

Scrolling in Vue Spreadsheet component

Scrolling helps you to move quickly to different areas of the worksheet. It moves faster if we use horizontal and vertical scroll bars. Scrolling can be enabled by setting the allowScrolling as true.

By default, the allowScrolling property is true.

You have the following options in Scrolling by using scrollSettings.

- Finite scrolling.
- Virtual scrolling.

Finite Scrolling

Finite scrolling supports two type of modes in scrolling. You can use the isFinite property in scrollSettings to specify the mode of scrolling.

- Finite - This mode does not create a new row/column when the scrollbar reaches the end. This can be achieved by setting the isFinite property as true.
- Infinite - This mode creates a new row/column when the scrollbar reaches the end. This can be achieved by setting the isFinite property as false.

By Default, the isFinite property is false.

Virtual Scrolling

- Virtual scrolling allows you to load data that you require (load data based on viewport size) without buffering the entire huge database. You can set the enableVirtualization property in scrollSettings as true.

In virtual scrolling enableVirtualization is set to true means, it allows you to load the spreadsheet data while scrolling.

By Default, the enableVirtualization property is true.

User Interface:

You can scroll through the worksheet using one of the following ways,

- Using the arrow keys.
- Using the Horizontal and Verticalscroll bars.
- Using the mouse wheel.

Finite scrolling with defined rows and columns

If you want to perform scrolling with defined rows and columns, you must define rowCount and colCount in the sheets property and set isFinite as true and enableVirtualization as false in scrollSettings.

The following code example shows the finite scrolling with defined rows and columns in the spreadsheet. Here, we used rowCount as 20 and colCount as 20, after reaching the 20th row or 20th column you can't able to scroll.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created"
:allowScrolling="true" :scrollSettings="scrollSettings">
                <e-sheets>
                  <e-sheet name="Price Details" rowCount="9" colCount="7">
                    <e-ranges>
                      <e-range :dataSource="dataSource1"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                    </e-columns>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { priceData  } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
        scrollSettings: { isFinite: true },
      dataSource1: priceData
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/spreadsheet/scrolling-cs1" %}

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

Protect sheet in Vue Spreadsheet component

Sheet protection helps you to prevent the users from modifying the data in the spreadsheet.

Protect Sheet

Protect sheet feature helps you to prevent the unknown users from accidentally changing, editing, moving, or deleting data in a spreadsheet.

You can use the isProtected property to enable or disable the Protecting functionality.

* The default value for isProtected property is false.

By default in protected sheet, selecting, formatting, inserting, deleting functionalities are disabled. To enable some of the above said functionalities the protectSettings options are used in a protected spreadsheet.

The available protectSettings options in spreadsheet are,

| Options | Uses |

|-----|------|

| Select Cells | Used to perform Cell Selection. |

| Format Cells | Used to perform Cell formatting. |

| Format Rows | Used to perform Row formatting. |

| Format Columns | Used to perform Column formatting. |

| Insert Link | Used to perform Hyperlink Insertions. |

* The default value for all protectSettings options are false.

By default, the Protect Sheet module is injected internally into the Spreadsheet to perform sheet protection function.

User Interface:

In the active Spreadsheet, the sheet protection can be done by any of the following ways:

- Select the Protect Sheet item in the Ribbon toolbar under the Data Tab, and then select your desired options.
- Right-click the sheet tab, select the Protect Sheet item in the context menu, and then select your desired options.
- Use the protectSheet() method programmatically.

The following example shows Protect Sheet functionality in the Spreadsheet control.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created">
                <e-sheets>
                  <e-sheet name="Budget" :isProtected="true"
:protectSettings="{ selectCells: true }">
                    <e-ranges>
                      <e-range :dataSource="dataSource1"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                    </e-columns>
                  </e-sheet>
                  <e-sheet name="Salary">
                    <e-ranges>
                      <e-range :dataSource="dataSource2"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      </e-columns>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { salaryData, budgetData  } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource1: budgetData,
      dataSource2: salaryData,
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
     spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:D1');
        spreadsheet.cellFormat({ fontWeight: 'bold'}, 'A11:D11');
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'Salary!A1:D1');
        spreadsheet.protectSheet(1, { selectCells: false});
      }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/protect-sheet-cs1" %}

Unprotect Sheet

Unprotect sheet is used to enable all the functionalities that are already disabled in a protected spreadsheet.

User Interface:

In the active Spreadsheet, the sheet Unprotection can be done by any of the following ways:

- Select the Unprotect Sheet item in the Ribbon toolbar under the Data Tab.
- Right-click the sheet tab, select the Unprotect Sheet item in the context menu.
- Use the unprotectSheet() method programmatically.

Unlock the particular cells in the protected sheet

In protected spreadsheet, to make some particular cell or range of cells are editable, you can use lockCells() method, with the parameter range and isLocked property as false.

APP.VUE

```
<template>
 <div> <ejs-button id='customBtn' v-on:click.native="btnClick"> Unlock
cells</ejs-button>
 <div id="dialog"></div>
              <ejs-spreadsheet ref="spreadsheet" id="spreadsheet"
:created="created">
                <e-sheets>
                  <e-sheet name="Budget" :isProtected="true"
:protectSettings="{ selectCells: true }">
                    <e-ranges>
                      <e-range :dataSource="dataSource1"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                    </e-columns>
                  </e-sheet>
                  <e-sheet name="Salary">
                    <e-ranges>
                      <e-range :dataSource="dataSource2"></e-range>
                    </e-ranges>
                    <e-columns>
```

```
                        <e-column :width=100></e-column>
                        <e-column :width=100></e-column>
                        <e-column :width=100></e-column>
                        <e-column :width=100></e-column>
                        </e-columns>
                    </e-sheet>
                  </e-sheets>
               </ejs-spreadsheet>
               </div>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { budgetData,salaryData } from './data.js';
import { Dialog } from '@syncfusion/ej2-popups';
Vue.use(SpreadsheetPlugin);
Vue.use(ButtonPlugin);
export default {
   data: () => {
    return {
       dialogObj: Dialog,
      dataSource1: budgetData,
      dataSource2: salaryData,
       dlgButtons: [{ click: this.lockCells, buttonModel: {
isPrimary:'true', content: 'Learn More' } }],
    }
  },
   methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
     spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:D1');
       spreadsheet.cellFormat({ fontWeight: 'bold'}, 'A11:D11');
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'Salary!A1:D1');
        // Creating dialog component,
        var dialogObj = new Dialog({
          header: 'Spreadsheet',
          target: document.getElementById('spreadsheet'),
          content: '"A1:F3" range of cells has been unlocked.',
          showCloseIcon: true,
          isModel: true,
          visible: false,
          width: '500px',
          buttons: [{
              click: this.lockCells.bind(this), buttonModel: { content:
'Ok', isPrimary: true }
          }]
      });
      dialogObj.appendTo('#dialog');
      },
      btnClick: function() {
        var dialogObj= document.getElementById("dialog").ej2_instances[0];
        dialogObj.show();
   },
   lockCells: function() {
```

```
        var spreadsheet = this.$refs.spreadsheet;
        var dialogObj= document.getElementById("dialog").ej2_instances[0];
        spreadsheet.lockCells('A1:F3', false);
        dialogObj.hide();
    }
    }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/lock-cells-cs1" %}

Limitations of Protect Sheet

- Password protection is not supported in Protect sheet feature.

Protect Workbook

Protect workbook feature helps you to protect the workbook so that users cannot insert, delete, rename, hide the sheets in the spreadsheet.

You can use the password property to protect workbook with password.

You can use the isProtected property to protect or unprotect the workbook without the password.

The default value for isProtected property is false.

User Interface:

In the active Spreadsheet, you can protect the worksheet by selecting the Data tab in the Ribbon toolbar and choosing the Protect Workbook item. Then, enter the password and confirm it and click on OK.

The following example shows Protect Workbook by using the isProtected property in the Spreadsheet control.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :isProtected="true">
   <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
          </e-sheet>
```

```
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: defaultData,
    }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/local-data-binding-cs4" %}

The following example shows Protect Workbook by using the password property in the Spreadsheet control. To unprotect the workbook, click the unprotect workbook button in the data tab and provide the password as syncfusion in the dialog box.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" password='syncfusion'>
   <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: defaultData,
```

```
    }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/local-data-binding-cs5" %}

Unprotect Workbook

Unprotect Workbook is used to enable the insert, delete, rename, move, copy, hide or unhide sheets feature in the spreadsheet.

User Interface:

In the active Spreadsheet, the workbook Unprotection can be done in any of the following ways:

- Select the Unprotect Workbook item in the Ribbon toolbar under the Data Tab and provide the valid password in the dialog box.

Limitations of Protect Workbook

- Encryption with password is not supported in the Protect workbook feature.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Hyperlink

Searching in Vue Spreadsheet component

Find and Replace helps you to search for the target text and replace the found text with alternative text within the sheet or workbook. You can use the allowFindAndReplace property to enable or disable the Find and Replace functionality.

* The default value for allowFindAndReplace property is true.

Copyright © 2001 -2024 Syncfusion Inc.

Find

Find feature is used to select the matched contents of a cell within the sheet or workbook. It is extremely useful when working with large set of data source.

User Interface:

Find can be done by any of the following ways:

- Select the Search icon in the Ribbon toolbar or use Ctrl + F key to open the Find dialog.
- Use find Next and find Previous buttons to search the given value in the workbook.
- Select the option button in Find dialog to open the Find and Replace dialog. Then, select the below properties for enhanced searching.

* Search within: To search the target in a sheet (default) or in an entire workbook.

* Search by: It enhance the search, either By Rows (default), or By Columns.

* Match case: To find the matched value with case sensitive.

* Match exact cell contents: To find the exact matched cell value with entire match cases.

- Using find() method to perform find operation.

Replace

Replace feature is used to change the find contents of a cell within sheet or workbook. Replace All is used to change all the matched contents of a cell within sheet or workbook.

User Interface:

Replace can be done by any of the following ways:

- Use Ctrl + H key to open the Find and Replace dialog.
- Use Replace button to change the found value in sheet or workbook.
- Using Replace All button, all the matched criteria can be replaced with find value based on sheet or workbook.
- Using replace() method to perform replace operation by passing the argument args.replaceby as replace.
- Using replace() method to perform replace all operation by passing the argument args.replaceby as replaceall.

Go to

Go to feature is used to navigate to a specific cell address in the sheet or workbook.

User Interface:

- Use Ctrl + G key to open the Go To dialog.
- Use goTo() method to perform Go To operation.

In the following sample, searching can be done by following ways:

- Select the Home tab in the Ribbon toolbar, and then select the Search icon.

Copyright © 2001 -2024 Syncfusion Inc.

- Enter any value in the search textbox.
- Select the next (or) previous button to find the entered value in the spreadsheet.
- You can have more options to find values by selecting the more options in the search toolbar.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :created="created"
:showFormulaBar="false">
                <e-sheets>
                  <e-sheet name="Price Details">
                    <e-ranges>
                      <e-range :dataSource="dataSource1"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                    </e-columns>
                  </e-sheet>
                  <e-sheet name="Budget Details">
                    <e-ranges>
                      <e-range :dataSource="dataSource2"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      </e-columns>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { priceData, budgetData  } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource1: priceData,
      dataSource2: budgetData,
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'Budget Details!A1:D1');
      }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/searching-cs1" %}

Limitations

- Undo/redo for Replace All is not supported in this feature.

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

Accessibility in Vue Spreadsheet component

The Spreadsheet component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Spreadsheet component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/not-supported.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Spreadsheet component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Spreadsheet component:

| Attributes | Purpose |

|-------|---------|

| grid (role) | This role is added to the spreadsheet content table and describes the collection of rows and columns. |

| gridcell (role) | This role is added to the cell element and describes the rows <td> element. |

| rowheader (role) | This role is added to the row header and describes the header of the rows. |

| colheader (role) | This role is added to the column header and describes the header of the columns. |

| aria-rowindex (attribute) | This attribute describes the table's row index in the spreadsheet. |

| aria-colindex (attribute) | This attribute describes the table's column index in the spreadsheet. |

| aria-selected (attribute) | This attribute describes an item's (cell, menu, checkbox, etc.) current selected state in the spreadsheet. |

| aria-rowcount (attribute) | This attribute describes the total number of rows in the table. |

| aria-colcount (attribute) | This attribute describes the total number of columns in the table. |

| aria-busy (attribute) | This attribute describes a currently updated or modified element. |

| aria-label (attribute) | This attribute describes the accessible name for the interactive elements. |

| textbox (role) | This role is assigned to the textbox that accepts text input. |

Copyright © 2001 -2024 Syncfusion Inc.

| menu (role) | This role has been added to the menu and describes the menu items. |

| aria-expanded (attribute) | This attribute describes the control (for example, dropdown) is expanded or collapsed. |

| aria-multiline (attribute) | This attribute defines what the Alt + Enter key does in the spreadsheet editor. |

Keyboard interaction

The Spreadsheet component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Spreadsheet component.

| Press | To do this |

|----------|-------------|

| Up arrow | Navigate from the active cell to the previous cell in the same column. |

| Down arrow | Navigate from the active cell to the next cell in the same column. |

| Left arrow | Navigate from the active cell to the previous cell in the same row. |

| Right arrow | Navigate from the active cell to the next cell in the same row. |

| Tab | Navigate the active cell to the next cell in the same row. |

| Shift + Tab | Navigate the active cell to the previous cell in the same row. |

| Home | Moves the selection to starting column in worksheet. |

| Ctrl + Home | Move the selection to the first visible cell on a worksheet. |

| Shift + Home | Extend the cell selection to the first column of a worksheet. |

| Ctrl + Shift + Home | Extend the selection of cells to the beginning of the worksheet. |

| Ctrl + End | Move to the last cell on a worksheet, right most last column and last row cell. |

| Page Up | Move page up. |

| Page Down | Move page down. |

| Shift + Page Up | Perform page up by selecting all cells between. |

| Shift + Page Down | Perform page down by selecting all cells between. |

| Ctrl + Up | Navigate to the non-blank cell before the active cell in the same column. |

| Ctrl + Down | Navigate to the last non-blank cell in the same column as the active cell. |

| Ctrl + Left | Navigate to the non-blank cell before the active cell in the same row. |

| Ctrl + Right | Navigate to the last non-blank cell in the same row as the active cell. |

| Shift + Up | Extend the selection of cells to the previous row. |

| Shift + Down | Extend the selection of cells to the next row. |

| Shift + Left | Extend the selection of cells to the previous column. |

| Shift + Right | Extend the selection of cells to the next column. |

Copyright © 2001 -2024 Syncfusion Inc.

| Ctrl + Shift + Up | Extend the cell selection to the previous non-blank cell in the same column as the active cell. |

| Ctrl + Shift + Down | Extend the cell selection to the last non-blank cell in the same column as the active cell. |

| Ctrl + Shift + Left | Extend the cell selection to the previous non-blank cell in the same row as the active cell. |

| Ctrl + Shift + Right | Extend the cell selection to the last non-blank cell in the same row as the active cell. |

| Enter | Navigate the active cell to the next cell in the same column. |

| Shift + Enter | Navigate to the previous cell in the same column from the active cell. |

| Alt + Enter | While editing, add a new line. |

| Enter | Complete the cell editing and select the cell below in the same column. |

| Shift + Enter | Complete the cell editing and select the cell above in the same column. |

| Tab | Complete the cell editing and select the next cell in the same row. |

| Shift + Tab | Complete the cell editing and select the previous cell in the same row. |

| Alt | Focus on the active ribbon tab. |

| Left | Move the focus to the previous items in the ribbon content. |

| Right | Move the focus to the next items in the ribbon content. |

| Alt + Down | Open the ribbon dropdown menu. |

| Esc / Alt + Up | Close the ribbon dropdown menu. |

Ensuring accessibility

The Spreadsheet component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Spreadsheet component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Spreadsheet component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/spreadsheet.html" %}

See also

- Accessibility in Syncfusion Vue components

Keyboard shortcuts in Vue Spreadsheet component

The keyboard shortcuts supported in the spreadsheet are,

| Shortcut | Description |

|-------|---------|

| Ctrl + O | Displays dialog to open a file. |

| Ctrl + S / Alt + F2 | Saves the workbook. |

Copyright © 2001 -2024 Syncfusion Inc.

| F2 | Enables edit mode. |

| ESC | Cancel edit mode and discard the changes. |

| Backspace and SPACE | Clears content of the active cell and enables edit mode. |

| Ctrl + C | Copies the selected cells. |

| Ctrl + X | Cuts the selected cells. |

| Ctrl + V | Paste the clipboard(cut or copied) content in the new selected range. |

| Ctrl + B | Applies or removes **bold** formatting. |

| Ctrl + I | Applies or removes *italic* formatting. |

| Ctrl + U | Applies or removes underline. |

| Ctrl + 5 | Applies or removes ~~strikethrough~~. |

| Ctrl + Z | Reverses (Undo) the last action. |

| Ctrl + Y | Repeats (Redo) the last reversed action. |

| Ctrl + K | It opens the Insert Hyperlink dialog for adding new hyperlink to a cell. If the selected cell already contains hyperlink, it opens the Edit Hyperlink dialog. |

| Ctrl + F / Shift + F5 | Opens Find dialog. |

| Ctrl + H | Opens Find and Replace dialog. |

| Ctrl + G | Opens GoTo dialog, which helps to navigate to cell. |

| Ctrl + Shift + L | Applies filter to the first row of the selected range or used range. |

| Alt + F | Opens the File menu. |

| Alt + H | Go to Home tab. |

| Alt + N | Go to Insert tab. |

| Alt + M | Go to Formulas tab. |

| Alt + A | Go to Data tab. |

| Alt + W | Go to View tab. |

| Tab | Navigate the active cell to the next cell in the same row. |

| Shift + Tab | Navigate the active cell to the previous cell in the same row. |

| Right or Left arrow| Move the focus to next or previous item in the tab if the focus is on ribbon tab.|

| Up arrow | When a menu is open, move focus to the next item. |

| Down arrow | When a menu is open, move focus to the previous item. |

| Spacebar or Enter | Activate a selected button. |

| Ctrl + F8 | Expand or collapse the ribbon content. |

| Ctrl + Shift + U | Expand or collapse the formula bar. |

| Ctrl + 9 | Hide the selected row. |

Copyright © 2001 -2024 Syncfusion Inc.

| Ctrl + 0 | Hide the selected column. |

| Home | Moves the selection to starting column in worksheet. |

| Ctrl + Home | Move the selection to the first visible cell on a worksheet. |

| Ctrl + Shift + Home | Extend the selection of cells to the beginning of the worksheet. |

| Ctrl + End | Move to the last cell on a worksheet, right most last column and last row cell. |

| Ctrl + & | Apply an outline border to the selected cells. |

| Ctrl + Shift + & | Apply an outline border to the cells that you've chosen. |

| Ctrl + Shift + ~ | Apply the General number format. |

| Ctrl + Shift + $ | Apply the Currency format with two decimal places (negative numbers in parentheses). |

| Shift + F10 | Open the context menu. |

| Ctrl + % | Apply the Percentage format with no decimal places. |

| Ctrl + ^ | Apply the Scientific number format with two decimal places. |

| Ctrl + Shift + # |Apply the Date format with the day, month, and year. |

| Ctrl + Shift + @ | Apply the Time format with the hour and minute, and AM or PM. |

| Ctrl + Shift + ! | Apply the Number format with two decimal places, thousands separator, and minus sign (-) for negative values. |

| Ctrl + Spacebar | Select an entire column in a worksheet. |

| Shift + Spacebar | Select an entire row in a worksheet. |

| Shift + F3 | Opens Insert Function dialog. |

| Ctrl + Alt + N | Opens new workbook. |

| Shift + Page Down | Perform page down by selecting all cells between. |

| Shift + Page Up | Perform page up by selecting all cells between. |

| Ctrl + Left | Navigate to the non-blank cell before the active cell in the same row. |

| Ctrl + Right | Navigate to the last non-blank cell in the same row as the active cell. |

| Ctrl + Up | Navigate to the first non-blank cell in the same row as the active cell. |

| Ctrl + Down | Navigate to the last cell that is not blank in the same column as the active cell. |

| Ctrl + Shift + Left | Extend the cell selection to the previous non-blank cell in the same row as the active cell. |

| Ctrl + Shift + Right | Extend the cell selection to the last non-blank cell in the same row as the active cell. |

| Ctrl + Shift + Up | Extend the selection of cells to the first non-blank cell in the same row as the active cell. |

| Ctrl + Shift + Down | Extend the selection of cells to the last non-blank cell in the same row as the active cell. |

Copyright © 2001 -2024 Syncfusion Inc.

| Shift + Alt + K | Open the List All Sheets dropdown option. |

| Up arrow | Navigate from the active cell to the previous cell in the same column. |

| Down arrow | Navigate from the active cell to the next cell in the same column. |

| Left arrow | Navigate from the active cell to the previous cell in the same row. |

| Right arrow | Navigate from the active cell to the next cell in the same row. |

| Page Up | Move page up. |

| Page Down | Move page down. |

| Shift + Up | Extend the selection of cells to the previous row. |

| Shift + Down | Extend the selection of cells to the next row. |

| Shift + Left | Extend the selection of cells to the previous column. |

| Shift + Right | Extend the selection of cells to the next column. |

| Ctrl + Top | Navigate to the non-blank cell before the active cell in the same column. |

| Ctrl + Shift + Top | Extend the cell selection to the previous non-blank cell in the same column as the active cell. |

| Ctrl + Shift + Bottom | Extend the cell selection to the last non-blank cell in the same column as the active cell. |

| Enter | Navigate the active cell to the next cell in the same column. |

| Shift + Enter | Navigate to the previous cell in the same column from the active cell. |

| Alt + Down | Open the list data validation menu and filter menu. |

| Alt + Up | Close the list data validation menu and filter menu. |

| Delete | Remove the contents of the cell. |

| Shift + Home | Extend the cell selection to the first column of a worksheet. |

| Shift + F11 | Add a new sheet. |

| Ctrl + Shift + 9 | Unhide the selected rows. |

| Ctrl + Shift + 0 | Unhide the selected columns. |

| Ctrl + D | Fill the cell down. |

| Ctrl + R | Fill the cell right. |

| Alt + Enter | While editing, add a new line. |

| Enter | Complete the cell editing and select the cell below in the same column. |

| Shift + Enter | Complete the cell editing and select the cell above in the same column. |

| Tab | Complete the cell editing and select the next cell in the same row. |

| Shift + Tab | Complete the cell editing and select the previous cell in the same row. |

| Alt | Focus on the active ribbon tab. |

| Left | Move the focus to the previous items in the ribbon content. |

Copyright © 2001 -2024 Syncfusion Inc.

| Right | Move the focus to the next items in the ribbon content. |

| Alt + Down | Open the ribbon dropdown menu. |

| Esc / Alt + Up | Close the ribbon dropdown menu. |

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Formatting

Ribbon in Vue Spreadsheet component

It helps to organize a spreadsheet's features into a series of tabs. By clicking the expand or collapse icon, you can expand or collapse the ribbon toolbar dynamically.

Ribbon Customization

You can customize the ribbon using the following methods,

| Method | Action |

|-------|---------|

| hideRibbonTabs | Used to show or hide the existing ribbon tabs. |

| enableRibbonTabs | Used to enable or disable the existing ribbon tabs. |

| addRibbonTabs | Used to add custom ribbon tabs. |

| hideToolbarItems | Used to show or hide the existing ribbon toolbar items. |

| enableToolbarItems | Used to enable or disable the specified toolbar items. |

| addToolbarItems | Used to add the custom items in ribbon toolbar. |

| hideFileMenuItems | Used to show or hide the ribbon file menu items. |

| enableFileMenuItems | Used to enable or disable file menu items. |

| addFileMenuItems | Used to add custom file menu items. |

The following code example shows the usage of ribbon customization.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :created="created"
:fileMenuBeforeOpen="fileMenuBeforeOpen"
                :fileMenuItemSelect="fileMenuItemSelect" :openUrl="openUrl"
:saveUrl="saveUrl" :showFormulaBar="false" :showSheetTabs="false"
:allowInsert="false" :allowDelete="false" :allowMerge="false">
                <e-sheets>
                  <e-sheet>
                    <e-ranges>
                      <e-range :dataSource="dataSource"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=180></e-column>
```

```
                        <e-column :width=130></e-column>
                        <e-column :width=130></e-column>
                        <e-column :width=180></e-column>
                        <e-column :width=130></e-column>
                        <e-column :width=120></e-column>
                    </e-columns>
                </e-sheet>
            </e-sheets>
        </ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
   data: () => {
    return {
      dataSource: data,
      openUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/open',
      saveUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/save',
    }
  },
  methods: {
  created: function () {
      var spreadsheet = this.$refs.spreadsheet;
       spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:F1');
        // Hiding the `Insert` from ribbon.
        spreadsheet.hideRibbonTabs(['Insert']);
        // Set disabled state to `View` ribbon tab.
        spreadsheet.enableRibbonTabs(['View'], false);
        /// Adding the `Help` ribbon tab at the last index.
        // Specify the ribbon tab header text in last optional
argument(`insertBefore`) for inserting new tab before any existing tab.
        spreadsheet.addRibbonTabs([{ header: { text: 'Help' }, content: [{
text: 'Feedback', tooltipText: 'Feedback',
            click : function() { /* Any click action for this toolbar item
will come here. */ } }] }]);
        // Hiding the unwanted toobar items from `Home` by specifying its
index.
        spreadsheet.hideToolbarItems('Home', [0, 1, 2, 4, 14, 15, 21, 24]);
        // Set diable state to `Underline`, 'Wrap text` toobar items from
`Home` by specifying the item id.
        spreadsheet.enableToolbarItems(
            'Home', [`${spreadsheet.ej2Instances.element.id}_underline`,
`${spreadsheet.ej2Instances.element.id}_wrap`], false);
        // Set disable state to `Protect Sheet` toolbar item from `Data` by
mentioning its index.
        spreadsheet.enableToolbarItems('Data', [0], false);
        // Adding the new `Custom Formulas` toolbar item under `Formulas`
tab for adding custom formulas.
        spreadsheet.addToolbarItems(
            'Formulas', [{ type: 'Separator' }, {
                text: 'Custom Formulas', tooltipText: 'Custom Formulas',
```

```
                // You can set click handler for each new custom toolbar
item.
                click: function() {
                    // You can add custom formualas here.
                }
            }]);
        // Adding new custom item `Import` after the existing `Open` item.
By default, new item will add after the specified item.
        spreadsheet.addFileMenuItems([{ text: 'Import', iconCss: 'e-open e-
icons' }], 'Open');
        // Adding new custom item `Export As` after the existing `Save As`
item.
        // Set `insertAfter` optional argument as `false` for adding new
item before the specified item.
        spreadsheet.addFileMenuItems(
            [{
                text: 'Export As', iconCss: 'e-save e-icons', items: [{
text: 'XLSX', iconCss: 'e-xlsx e-icons' },
                    { text: 'XLS', iconCss: 'e-xls e-icons' }, { text:
'CSV', iconCss: 'e-csv e-icons' }]
            }],
            'Save As', false);
      },
       fileMenuBeforeOpen: function () {
           var spreadsheet = this.$refs.spreadsheet;
        // Because the file menu items are created dynamically, you need to
perform the hide or show and enable/disable operations
        // under filemenu before open event.
        // Hiding the `Save As` and `Open` item.
        spreadsheet.hideFileMenuItems(['Save As', 'Open']);
         // Set disable state to `New` item. You can perform any file menu
items customization by specifying the item id,
        // if it has more than one same item text. Set the last argument
`isUniqueId` as true for using the item id.

spreadsheet.enableFileMenuItems([`${spreadsheet.ej2Instances.element.id}_New
`], false, true);
    },
    fileMenuItemSelect : function (args) {
        var spreadsheet = this.$refs.spreadsheet;
        // Custom file menu items select handler.
        switch (args.item.text) {
            case 'Import': select(`#${spreadsheet.element.id}_fileUpload`,
spreadsheet.element).click();
                break;
            case 'XLSX': spreadsheet.save({ saveType: 'Xlsx' });
                break;
            case 'XLS': spreadsheet.save({ saveType: 'Xls' });
                break;
            case 'CSV': spreadsheet.save({ saveType: 'Csv' });
                break;
      }
    }
    }
}
</script>
<style>
```

```
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/ribbon-cs1" %}

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Worksheet
- Rows and columns

Global local in Vue Spreadsheet component

Localization

The Localization library allows you to localize the default text content of the Spreadsheet. The Spreadsheet has static text on some features (cell formatting, Merge, Data validation, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the

locale value and translation object.

The following list of properties and their values are used in the Spreadsheet.

Locale keywords |Text

Cut | Cut

Copy | Copy

Paste | Paste

PasteSpecial | Paste Special

All | All

Values | Values

Formats | Formats

Font | Font

FontSize | Font Size

Bold | Bold

Italic | Italic

Copyright © 2001 -2024 Syncfusion Inc.

Underline | Underline

Strikethrough | Strikethrough

TextColor | Text Color

FillColor | Fill Color

HorizontalAlignment | Horizontal Alignment

AlignLeft | Align Left

AlignCenter | Center

AlignRight | Align Right

VerticalAlignment | Vertical Alignment

AlignTop | Align Top

AlignMiddle | Align Middle

AlignBottom | Align Bottom

InsertFunction | Insert Function

Insert | Insert

Delete | Delete

Rename | Rename

Hide | Hide

Unhide | Unhide

NameBox | Name Box

ShowHeaders | Show Headers

HideHeaders | Hide Headers

ShowGridLines | Show Gridlines

HideGridLines | Hide Gridlines

AddSheet | Add Sheet

ListAllSheets | List All Sheets

FullScreen | Full Screen

CollapseToolbar | Collapse Toolbar

ExpandToolbar | Expand Toolbar

CollapseFormulaBar | Collapse Formula Bar

ExpandFormulaBar | Expand Formula Bar

File | File

Home | Home

Formulas | Formulas

Copyright © 2001 -2024 Syncfusion Inc.

View | View

New | New

Open | Open

SaveAs | Save As

ExcelXlsx | Microsoft Excel

ExcelXls | Microsoft Excel 97-2003

CSV | Comma-separated values

FormulaBar | Formula Bar

Ok | Ok

Close | Close

Cancel | Cancel

Apply | Apply

MoreColors | More Colors

StandardColors | Standard Colors

General | General

Number | Number

Currency | Currency

Accounting | Accounting

ShortDate | Short Date

LongDate | Long Date

Time | Time

Percentage | Percentage

Fraction | Fraction

Scientific | Scientific

Text | Text

NumberFormat | Number Format

MobileFormulaBarPlaceHolder | Enter value or Formula

PasteAlert | You can't paste it here because the copy area and paste area aren't in the same size. Please try pasting in a different range.

DestroyAlert | Are you sure you want to destroy the current workbook without saving and create a new workbook?

SheetRenameInvalidAlert | Sheet name contains invalid character.

SheetRenameEmptyAlert | Sheet name cannot be empty.

SheetRenameAlreadyExistsAlert | Sheet name already exists. Please enter another name.

Copyright © 2001 -2024 Syncfusion Inc.

DeleteSheetAlert | Are you sure you want to delete this sheet?

DeleteSingleLastSheetAlert | A Workbook must contain at least one visible worksheet.

PickACategory | Pick a category

Description | Description

UnsupportedFile | Unsupported File

InvalidUrl | Invalid Url

SUM | Adds a series of numbers and/or cells.

SUMIF | Adds the cells based on the specified condition.

SUMIFS | Adds the cells based on the specified conditions.

ABS | Returns the value of a number without its sign.

RAND | Returns a random number between 0 and 1.

RANDBETWEEN | Returns a random integer based on the specified values.

FLOOR | Rounds a number down to the nearest multiple of a given factor.

CEILING | Rounds a number up to the nearest multiple of a given factor.

PRODUCT | Multiplies a series of numbers and/or cells.

AVERAGE | Calculates average for the series of numbers and/or cells excluding text.

AVERAGEIF | Calculates average for the cells based on the specified criterion.

AVERAGEIFS | Calculates average for the cells based on the specified conditions.

AVERAGEA | Calculates the average for the cells evaluating TRUE as 1 text and FALSE as 0.

COUNT | Counts the cells that contain numeric values in a range.

COUNTIF | Counts the cells based on the specified condition.

COUNTIFS | Counts the cells based on specified conditions.

COUNTA | Counts the cells that contain values in a range.

MIN | Returns the smallest number of the given arguments.

MAX | Returns the largest number of the given arguments.

DATE | Returns the date based on the given year, month, and day.

DAY | Returns the day from the given date.

DAYS | Returns the number of days between two dates.

IF | Returns value based on the given expression.

IFS | Returns value based on the given multiple expressions.

AND | Returns TRUE if all the arguments are TRUE otherwise returns FALSE.

OR | Returns TRUE if any of the arguments are TRUE otherwise returns FALSE.

IFERROR | Returns value if no error found else it will return specified value.

Copyright © 2001 -2024 Syncfusion Inc.

CHOOSE | Returns a value from the list of values based on the index number.

INDEX | Returns the value of the cell in a given range based on row and column number.

FIND | Returns the position of a string within another string which is case sensitive.

CONCATENATE | Combines two or more strings together.

CONCAT | Concatenates a list or a range of text strings.

SUBTOTAL | Returns subtotal for a range using the given function number.

RADIANS | Converts degrees into radians.

MATCH | Returns the relative position of a specified value in the given range.

DefineNameExists | This name already exists try a different name.

CircularReference | When a formula refers to one or more circular references this may result in an incorrect calculation.

ShowRowsWhere | Show rows where |

CustomFilterDatePlaceHolder | Choose a date

CustomFilterPlaceHolder | Enter the value

CustomFilter | Custom Filter

Between | Between

MatchCase | Match Case

DateTimeFilter | DateTime Filters

Undo | Undo

Redo | Redo

DateFilter | Date Filters

TextFilter | Text Filters

NumberFilter | Number Filters

ClearFilter | Clear Filter

NoResult | No Matches Found

FilterFalse | False

FilterTrue | True

Blanks | Blanks

SelectAll | Select All

GreaterThanOrEqual | Greater Than Or Equal

GreaterThan | Greater Than

LessThanOrEqual | Less Than Or Equal

LessThan | Less Than

Copyright © 2001 -2024 Syncfusion Inc.

NotEqual | Not Equal

Equal | Equal

Contains | Contains

EndsWith | Ends With

StartsWith | Starts With

ClearButton | Clear

FilterButton | Filter

CancelButton | Cancel

OKButton | OK

Search | Search

Link | Link

Hyperlink | Hyperlink

EditHyperlink | Edit Hyperlink

OpenHyperlink | Open Hyperlink

RemoveHyperlink | Remove Hyperlink

InsertLink | Insert Link

EditLink | Edit Link

WebPage | WEB PAGE

ThisDocument | THIS DOCUMENT

DisplayText | Display Text

Url | URL

CellReference | Cell Reference

Sheet | Sheet

DefinedNames | Defined Names

EnterTheTextToDisplay | Enter the text to display

EnterTheUrl | Enter the URL

ProtectSheet | Protect Sheet

UnprotectSheet | Unprotect Sheet

SelectCells | Select cells

FormatCells | Format cells

FormatRows | Format rows

Format Columns | Format columns

InsertLinks | Insert links

Copyright © 2001 -2024 Syncfusion Inc.

ProtectContent | Protect the contents of locked cells

ProtectAllowUser | Allow all users of this worksheet to |

EditAlert | The cell you're trying to change is protected. To make a change, unprotect the sheet.

FindReplaceTooltip | Find & Replace

InsertingEmptyValue | Reference value is not valid.

ByRow | By Row

ByColumn | By Column

MatchExactCellElements | Match Exact Cell Contents

EntercellAddress | Enter Cell Address

FindAndReplace | Find and Replace

ReplaceAllEnd | matches replaced with.

FindNextBtn | Find Next

FindPreviousBtn | Find Previous

ReplaceBtn | Replace

ReplaceAllBtn | Replace All

GotoHeader | Go To

GotoSpecialHeader | Go To Special

SearchWithin | Search within

SearchBy | Search by

Reference | Reference

Workbook | Workbook

NoElements | We couldn’t find what you were looking for.

FindWhat | Find what

ReplaceWith | Replace with

EnterValue | Enter Value

DefineNameInValid | The name that you entered is not valid.

FindValue | Find Value

ReplaceValue | Replace Value

DataValidation | Data Validation

CLEARALL | CLEAR ALL

APPLY | APPLY

CellRange | Cell Range

Allow | Allow

Copyright © 2001 -2024 Syncfusion Inc.

Data | Data

Minimum | Minimum

Maximum | Maximum

IgnoreBlank | Ignore blank

WholeNumber | Whole Number

Decimal | Decimal

Date | Date

TextLength | Text Length

List | List

NotBetween | Not between

EqualTo | Equal to

NotEqualTo | Not equal to

Greaterthan | Greater than

Lessthan | Less than

GreaterThanOrEqaulTo | Greater than or eqaul to

LessThanOrEqualTo | Less than or equal to

InCellDropDown | In-cell-dropdown

Sources | Sources

Value | Value

Retry | Retry

DialogError | The list source must be a reference to a single row or column.

ValidationError | This value doesn't match the data validation restrictions defined for the cell.

HideRow | Hide Row

HideRows | Hide Rows

UnHideRows | UnHide Rows

HideColumn | Hide Column

HideColumns | Hide Columns

UnHideColumns | UnHide Columns

InsertRow | Insert Row

InsertRows | Insert Rows

InsertColumn | Insert Column

InsertColumns | Insert Columns

DeleteRow | Delete Row

Copyright © 2001 -2024 Syncfusion Inc.

DeleteRows | Delete Rows

DeleteColumn | Delete Column

DeleteColumns | Delete Columns

Borders | Borders

TopBorders | Top Borders

LeftBorders | Left Borders

RightBorders | Right Borders

BottomBorders | Bottom Borders

AllBorders | All Borders

HorizontalBorders | Horizontal Borders

VerticalBorders | Vertical Borders

OutsideBorders | Outside Borders

InsideBorders | Inside Borders

NoBorders | No Borders

BorderColor | Border Color

BorderStyle | Border Style

INTERCEPT | Calculates the point of the Y-intercept line via linear regression.

SLOPE | Returns the slope of the line from linear regression of the data points.

FreezePanes | Freeze Panes

FreezeRows | Freeze Rows

FreezeColumns | Freeze Columns

UnfreezePanes | Unfreeze Panes

UnfreezeRows | Unfreeze Rows

UnfreezeColumns | Unfreeze Columns

MergeCells | Merge Cells

MergeAll | Merge All

MergeHorizontally | Merge Horizontally

MergeVertically | Merge Vertically

Unmerge | Unmerge

UnmergeCells | Unmerge Cells

SortAscending | Ascending

SortDescending | Descending

CustomSort | Custom Sort

Copyright © 2001 -2024 Syncfusion Inc.

ClearAllFilter | Clear

ReapplyFilter | Reapply

Clear | Clear

ClearContents | Clear Contents

ClearAll | Clear All

ClearFormats | Clear Formats

ClearHyperlinks | Clear Hyperlinks

Image | Image

AddColumn | Add Column

SortBy | Sort by

ThenBy | Then by

Chart | Chart

Column | Column

Bar | Bar

Area | Area

Pie | Pie

Doughnut | Doughnut

PieAndDoughnut | Pie/Doughnut

Line | Line

Radar | Radar

Scatter | Scatter

ChartDesign | Chart Design

ClusteredColumn | Clustered Column

StackedColumn | Stacked Column

StackedColumn100 | 100% Stacked Column

ClusteredBar | Clustered Bar

StackedBar | Stacked Bar

StackedBar100 | 100% Stacked Bar

StackedArea | Stacked Area

StackedArea100 | 100% Stacked Area

StackedLine | Stacked Line

StackedLine100 | 100% Stacked Line

AddChartElement | Add Chart Element

Copyright © 2001 -2024 Syncfusion Inc.

Axes | Axes

AxisTitle | Axis Title

ChartTitle | Chart Title

DataLabels | Data Labels

Gridlines | Gridlines

Legends | Legends

PrimaryHorizontal | Primary Horizontal

PrimaryVertical | Primary Vertical

None | None

AboveChart | Above Chart

Center | Center

InsideEnd | Inside End

InsideBase | Inside Base

OutsideEnd | OutSide End

PrimaryMajorHorizontal | Primary Major Horizontal

PrimaryMajorVertical | Primary Major Vertical

PrimaryMinorHorizontal | Primary Minor Horizontal

PrimaryMinorVertical | Primary Minor Vertical

Right | Right

Left | Left

Bottom | Bottom

Top | Top

SwitchRowColumn | Switch Row/Column

ChartTheme | Chart Theme

ChartType | Chart Type

Material | Material

Fabric | Fabric

Bootstrap | Bootstrap

HighContrastLight | HighContrastLight

MaterialDark | MaterialDark

FabricDark | FabricDark

HighContrast | HighContrast

BootstrapDark | BootstrapDark

Copyright © 2001 -2024 Syncfusion Inc.

Bootstrap4 | Bootstrap4

VerticalAxisTitle | Vertical Axis Title

HorizontalAxisTitle | Horizontal Axis Title

EnterTitle | Enter Title

ProtectWorkbook | Protect Workbook

Password | Password (optional) |

unProtectPassword | Password

EnterThePassword | Enter the password

ConfirmPassword | Confirm Password

EnterTheConfirmPassword | Re-enter your password

PasswordAlert | Confirmation password is not identical

UnProtectWorkbook | Unprotect Workbook

UnProtectPasswordAlert | The password you supplied is not correct.

InCorrectPassword | Unable to open the file or worksheet with the given password.

PasswordAlertMsg | Please enter the password

ConfirmPasswordAlertMsg | Please enter the confirm password

IsProtected | is protected

Loading translations

To load translation object in an application, use load function of the L10n class.

The following example demonstrates the Spreadsheet in French culture. In the below sample we have translated the ribbon tab names and Home tab content (clipboard, cell style).

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" locale="fr-CH">
   <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { L10n } from '@syncfusion/ej2-base';
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
L10n.load({
    'fr-CH': {
        'spreadsheet': {
            'File': 'Fichier',
```

```
            'Home': 'Accueil',
            'Insert': 'Insérer',
            'Formulas': 'Formules',
            'Data': 'Les données',
            'View': 'Vue',
            'Cut': 'Coupe',
            'Copy': 'Copie',
            'Paste': 'Pâte',
            'PasteSpecial': 'Pâte spéciale',
            'All': 'Tous les',
            'Values': 'Valeurs',
            'Formats': 'Les formats',
            'Font': 'fonte',
            'FontSize': 'Taille de police',
            'Bold': 'Audacieux',
            'Italic': 'Italique',
            'Underline': 'Souligner',
            'Strikethrough': 'Barré',
            'TextColor': 'Couleur du texte',
            'FillColor': 'La couleur de remplissage',
            'HorizontalAlignment': 'Alignement horizontal',
            'AlignLeft': 'Alignez à gauche',
            'AlignCenter': 'centre',
            'AlignRight': 'Aligner à droite',
            'VerticalAlignment': 'Alignement vertical',
            'AlignTop': 'Aligner en haut',
            'AlignMiddle': 'Aligner le milieu',
            'AlignBottom': 'Aligner le bas',
            'InsertFunction': 'Insérer une fonction',
            'Delete': 'Effacer',
            'Rename': 'Rebaptiser',
            'Hide': 'Cacher',
            'Unhide': 'Démasquer',
            'NumberFormat': 'Nombre Format',
        }
    }
});
export default {
   data: () => {
    return {
      dataSource: defaultData,
    }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/local-data-binding-cs2" %}

Internationalization

The Internationalization library is used to globalize number, date, and time values in the spreadsheet component.

The following example demonstrates the Spreadsheet in French [fr-CH] culture. In the below sample we have globalized the Date(Date column), Time(Time column), and Currency(Amount column) formats.

`

```
<template>
<ejs-spreadsheet ref="spreadsheet" locale="fr-CH" :created="created">
<e-sheets>
<e-sheet>
<e-ranges>
<e-range :dataSource="dataSource"></e-range>
</e-ranges>
</e-sheet>
</e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import {
L10n,
setCulture,
setCurrencyCode,
loadCldr,
} from "@syncfusion/ej2-base";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { defaultData } from './data.js';
L10n.load({
'fr-CH': {
'spreadsheet': {
'File': 'Fichier',
'Home': 'Accueil',
```

```
'Insert': 'Insérer',
'Formulas': 'Formules',
'Data': 'Les données',
'View': 'Vue',
'Cut': 'Coupe',
'Copy': 'Copie',
'Paste': 'Pâte',
'PasteSpecial': 'Pâte spéciale',
'All': 'Tous les',
'Values': 'Valeurs',
'Formats': 'Les formats',
'Font': 'fonte',
'FontSize': 'Taille de police',
'Bold': 'Audacieux',
'Italic': 'Italique',
'Underline': 'Souligner',
'Strikethrough': 'Barré',
'TextColor': 'Couleur du texte',
'FillColor': 'La couleur de remplissage',
'HorizontalAlignment': 'Alignement horizontal',
'AlignLeft': 'Alignez à gauche',
'AlignCenter': 'centre',
'AlignRight': 'Aligner à droite',
'VerticalAlignment': 'Alignement vertical',
'AlignTop': 'Aligner en haut',
'AlignMiddle': 'Aligner le milieu',
'AlignBottom': 'Aligner le bas',
'InsertFunction': 'Insérer une fonction',
'Delete': 'Effacer',
'Rename': 'Rebaptiser',
'Hide': 'Cacher',
'Unhide': 'Démasquer',
'NumberFormat': 'Nombre Format',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
}
});
Vue.use(SpreadsheetPlugin);
import * as numberingSystems from "./numberingSystems.json";
import * as gregorian from "./ca-gregorian.json";
import * as numbers from "./numbers.json";
import * as timeZoneNames from "./timeZoneNames.json";
import * as currencies from "./currencies.json";
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames, currencies);
setCulture("fr-CH");
setCurrencyCode("EUR");
export default {
data: () => {
return {
dataSource: defaultData
}
},
methods: {
created: function () {
var spreadsheet = this.$refs.spreadsheet;
//Applies cell and number formatting to specified range of the active sheet
spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center', verticalAlign: 'middle' }, 'A1:F1');
spreadsheet.numberFormat('$#,##0.00', 'F2:F11');
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-spreadsheet/styles/material.css";
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';

@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';

@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';

@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';

@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';

@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";

</style>

`

Internalization sample link

Right to left (RTL)

RTL provides an option to switch the text direction and layout of the Spreadsheet component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL Spreadsheet, set the enableRtl to true.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" locale="ar-AE" :enableRtl="true">
   <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { L10n } from '@syncfusion/ej2-base';
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
L10n.load({
      'ar-AE': {
        'spreadsheet': {
            "File": "ملف",
            "Home": "هم",
            "Insert": "إدراج",
            "Formulas": "الصيغ",
            "View": "معاينة",
            "Data": "البيانات",
            "Cut": "قطع",
            "Copy": "نسخ",
            "Paste": "معجون",
            "PasteSpecial": "لصق خاص",
            "All": "جميع",
            "Values": "القيم",
            "Formats": "شكل",
            "Font": "الخط",
            "FontSize": "حجم الخط",
```

```
            "Bold": "جريء",
            "Italic": "مائل",
            "Underline": "أكد",
            "Strikethrough": "يتوسطه",
            "TextColor": "لون الخط",
            "FillColor": "لون التعبئة",
            "HorizontalAlignment": "المحاذاة الأفقية",
            "AlignLeft": "محاذاة إلى اليسار",
            "AlignCenter": "مركز",
            "AlignRight": "محاذاة إلى اليمين",
            "VerticalAlignment": "محاذاة عمودية",
            "AlignTop": "محاذاة أعلى",
            "AlignMiddle": "محاذاة الأوسط",
            "AlignBottom": "أسفل محاذاة",
            "InsertFunction": "إدراج وظيفة",
            "Delete": "حذف",
            "Rename": "إعادة تسمية",
            "Hide": "إخفاء",
            "Unhide": "ظهار"
        }
    }
});
export default {
   data: () => {
    return {
      dataSource: defaultData,
    }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/local-data-binding-cs3" %}

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Localization

Copyright © 2001 -2024 Syncfusion Inc.

Undo redo in Vue Spreadsheet component

Undo option helps you to undone the last action performed and Redo option helps you to do the same action which is reverted in the Spreadsheet. You can use the allowUndoRedo property to enable or disable undo redo functionality in spreadsheet.

* The default value for allowUndoRedo property is true.

By default, the UndoRedo module is injected internally into Spreadsheet to perform undo redo.

Undo

It reverses the last action you performed with Spreadsheet. Undo can be done by any of the following ways:

- Select the undo item from HOME tab in Ribbon toolbar.
- Use Ctrl + Z keyboard shortcut to perform the undo.
- Use the undo method programmatically.

Redo

It reverses the last undo action you performed with Spreadsheet. Redo can be done by any of the following ways:

- Select the redo item from HOME tab in Ribbon toolbar.
- Use Ctrl + Y keyboard shortcut to perform the redo.
- Use the redo method programmatically.

Update custom actions in UndoRedo collection

You can update your own custom actions in UndoRedo collection, by using the updateUndoRedoCollection method. And also customize the undo redo operations of your custom action by using actionComplete event.

The following code example shows How to update and customize your own actions for undo redo functionality in the Spreadsheet control.

APP.VUE

```
<template>
  <div>
    <ejs-button id='customBtn' v-
on:click.native="updateCollection">add/remove Class</ejs-button>
    <ejs-spreadsheet ref="spreadsheet" :actionComplete="actionComplete">
      <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
             <e-columns>
              <e-column :width="width1"></e-column>
              <e-column :width="width2"></e-column>
            </e-columns>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
        <div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin, getRangeIndexes } from "@syncfusion/ej2-vue-
spreadsheet";
import { addClass, removeClass } from '@syncfusion/ej2-base';
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
Vue.use(ButtonPlugin);
export default {
   data: () => {
    return {
      dataSource: defaultData,
      width1: 130,
      width2: 96,
    }
  },
  methods: {
   actionComplete: function (args) {
        var actionEvents = args;
         var spreadsheet = this.$refs.spreadsheet;
        if (actionEvents.eventArgs.action == "customCSS") {
            var Element =
spreadsheet.ej2Instances.getCell(actionEvents.eventArgs.rowIdx,actionEvents.
eventArgs.colIdx);
            if (actionEvents.eventArgs.requestType == "undo") {
                removeClass([Element],'customClass'); // To remove the
custom class in undo action
            }
            else {
                addClass([Element],'customClass');// To add the custom class
in redo action
            }
        }
    },
    updateCollection: function() {
    var spreadsheet = this.$refs.spreadsheet;
    var cell = spreadsheet.ej2Instances.getActiveSheet().activeCell;
    var cellIdx = getRangeIndexes(cell);
    var Element = spreadsheet.ej2Instances.getCell(cellIdx[0], cellIdx[1]);
    if (!Element.classList.contains("customClass")) {
        addClass([Element], 'customClass'); // To add the custom class in
active cell element
        spreadsheet.ej2Instances.updateUndoRedoCollection({ eventArgs: {
class: 'customClass', rowIdx: cellIdx[0], colIdx: cellIdx[1], action:
'customCSS' } }); // To update the undo redo collection
    }
   }
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
```

```
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
  .customClass.e-cell {
                      background-color: red;
                  }
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/undo-redo-cs1" %}

Note

You can refer to our Vue Spreadsheet feature tour page for its groundbreaking feature representations. You can also explore our Vue Spreadsheet example to knows how to present and manipulate data.

See Also

- Sorting
- Filtering
- Hyperlink

Use Cases

Collaborative editing in Vue Spreadsheet component

The collaborative editing support allows you to work at a spreadsheet collaboratively with other users. Multiple users can access to the the same spreadsheet simultaneously.

Dependencies

The following list of dependencies are required to use the collaborative editing support in spreadsheet.

`js

|-- @aspnet/signalr

|-- eventsource

|-- request

|-- ws

`

* Run the command `npm i @aspnet/signalr` to install `@aspnet/signalr` packages in your application.

Client configuration

To broadcast the data for every action in the spreadsheet, you need to transfer the data to the server through `send` method in `actionComplete` event and receive the same data by using the `dataReceived` method. In the `dataReceived` method, you need to update the action to the connected clients through `updateAction` method.

The following code example shows `Collaborative Editing` support in the Spreadsheet control.

`

```
<template>
<ejs-spreadsheet ref="spreadsheet" :actionComplete="actionComplete" :showFormulaBar="false">
<e-sheets>
<e-sheet>
<e-columns>
<e-column :width=180></e-column>
<e-column :width=130></e-column>
<e-column :width=130></e-column>
</e-columns>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import * as signalR from '@aspnet/signalr';
Vue.use(SpreadsheetPlugin);
export default {
data: () => {
return {
connection: null,
}
},
mounted: function () {
var spreadsheet = this.$refs.spreadsheet;
this.connection = new
signalR.HubConnectionBuilder().withUrl('https://localhost:44385/hubs/spreadsheethub', { // localhost
from AspNetCore service
skipNegotiation: true,
transport: signalR.HttpTransportType.WebSockets
}).build();
this.connection.start().then(() => { // to start the server
console.log('server connected!!!');
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}).catch(err => console.log(err));
this.connection.on('dataReceived', (data) => {
var model = JSON.parse(data);
spreadsheet.updateAction(model); // update the action to the connected clients
});
},
methods: {
actionComplete: function(args) {
this.connection.send('BroadcastData', JSON.stringify(args)); // send the action data to the server
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-spreadsheet/styles/material.css";
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
`
```

Server configuration

To make the communication between the server to the connected clients and from clients to the server, you need to configure the signalR Hubs using the following code.

```
`js
import * as signalR from '@aspnet/signalr';
// For signalR Hub connection
```

public connection = new signalR.HubConnectionBuilder().withUrl('https://localhost:44385/hubs/spreadsheethub', { // localhost from AspNetCore service

skipNegotiation: true,

transport: signalR.HttpTransportType.WebSockets

}).build();

`

Hub configuration

Initially create a AspNetCore project and add the hub file for sending and receiving the data between server and clients.

`javascript

using Microsoft.AspNetCore.SignalR;

using System.Threading.Tasks;

namespace WebApplication.Hubs

{

public class SpreadsheetHub : Hub

{

public async Task BroadcastData(string data)

{

await Clients.Others.SendAsync("dataReceived", data);

}

}

}

`

To configure the SignalR middleware by registering the following service in the ConfigureServices method of the Startup class.

`javascript

services.AddSignalR(e =>

{

e.MaximumReceiveMessageSize = int.MaxValue; // Option to increase message size for inserting image feature. By default, SignalR send messages up to 32 KB.

});

`

To set up the SignalR routes by calling MapHub in the Configure method of the Startup class.

`javascript

Copyright © 2001 -2024 Syncfusion Inc.

```
app.UseEndpoints(endpoints =>
{
endpoints.MapRazorPages();
endpoints.MapHub<SpreadsheetHub>("/hubs/spreadsheethub");
});
```

For hosting the service, you may use the above code snippet or download and run the local service.

Prevent the particular action update for collaborative client

Using the `action` argument from the `actionComplete` event, you can prevent the particular action update for collaborative client.

The following code example shows how to prevent collaborative client from updating the `format` action.

```
<template>
<ejs-spreadsheet ref="spreadsheet" :actionComplete="actionComplete" :showFormulaBar="false">
<e-sheets>
<e-sheet>
<e-columns>
<e-column :width=180></e-column>
<e-column :width=130></e-column>
<e-column :width=130></e-column>
</e-columns>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import * as signalR from '@aspnet/signalr';
Vue.use(SpreadsheetPlugin);
export default {
data: () => {
return {
connection: null,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
},
mounted: function () {
var spreadsheet = this.$refs.spreadsheet;
this.connection = new signalR.HubConnectionBuilder().withUrl('https://localhost:44385/hubs/spreadsheethub', { // localhost from AspNetCore service
skipNegotiation: true,
transport: signalR.HttpTransportType.WebSockets
}).build();
this.connection.start().then(() => { // to start the server
console.log('server connected!!!');
}).catch(err => console.log(err));
this.connection.on('dataReceived', (data) => {
var model = JSON.parse(data);
spreadsheet.updateAction(model); // update the action to the connected clients
});
},
methods: {
actionComplete: function(args) {
if (args.action != 'format'){
this.connection.send('BroadcastData', JSON.stringify(args)); // send the action data to the server
}
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-spreadsheet/styles/material.css";
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
```

@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';

@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';

@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';

@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";

</style>

`

Perform import action for collaborative clients

Using the action argument from the actionComplete event, you can identity whether the import action is performed or not. If the action is import, then you need to send the response data to the server and also update the same to the collaborative clients.

The following code example shows how to perform the import functionality for collaborative clients.

`

<template>

<ejs-spreadsheet ref="spreadsheet" :openUrl="openUrl" :actionComplete="actionComplete"></ejs-spreadsheet>

</template>

<script>

import { SpreadsheetComponent } from "@syncfusion/ej2-vue-spreadsheet";

import { isNullOrUndefined } from "@syncfusion/ej2-base";

import * as signalR from '@microsoft/signalr';

export default {

components: {

"ejs-spreadsheet": SpreadsheetComponent,

},

data: () => {

return {

connection: null,

openUrl: 'https://services.syncfusion.com/vue/production/api/spreadsheet/open',

}

},

mounted: function () {

// Fetch the spreadsheet instance.

var spreadsheet = this.$refs.spreadsheet;

this.connection = new signalR.HubConnectionBuilder().withUrl('https://localhost:44385/hubs/spreadsheethub', {

Copyright © 2001 -2024 Syncfusion Inc.

```
skipNegotiation: true,
transport: signalR.HttpTransportType.WebSockets
}).build();
this.connection.start().then(() => {
console.log('server connected!!!');
}).catch(err => console.log(err));
this.connection.on('dataReceived', (data) => {
const model = JSON.parse(
data
);
// Condition to check whether action performed is import.
if (isNullOrUndefined(model['action'])) {
// Load the imported excel file data as JSON to the connected clients.
const jsonData = { Workbook: model };
spreadsheet.openFromJson({ file: jsonData });
}
else {
// Update the action details to the connected clients.
spreadsheet.updateAction(model);
}
});
},
methods: {
actionComplete: function (args) {
if (args.action === 'import') {
// Send the action data to the server in args.response at the time of importing an excel file.
this.connection.send("BroadcastData", JSON.stringify(args.response.data));
}
else {
// Send the action data to the server for other than import actions.
this.connection.send("BroadcastData", JSON.stringify(args));
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

}

}

</script>

<style>

@import '../node_modules/@syncfusion/ej2-base/styles/material.css';

@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';

@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';

@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';

@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';

@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';

@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';

@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";

@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';

@import "../node_modules/@syncfusion/ej2-vue-spreadsheet/styles/material.css";

</style>

`

How To

Sort a range by custom list in Vue Spreadsheet component

You can also define the sorting of cell values based on your own customized personal list. In this article, custom list is achieved using custom sort comparer.

In the following demo, the Trustworthiness column is sorted based on the custom lists Perfect, Sufficient, and Insufficient.

APP.VUE

```
<template>
   <ejs-spreadsheet ref="spreadsheet" :allowSorting='true'
:dataBound="dataBound" :sortComplete="sortComplete">
   <e-sheets>
          <e-sheet>
            <e-ranges>
              <e-range :dataSource="dataSource"></e-range>
            </e-ranges>
          </e-sheet>
        </e-sheets></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { DataManager, DataUtil } from '@syncfusion/ej2-data';
import { tradeData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
```

```
    data: () => {
     return {
       dataSource: tradeData,
     }
   },
   methods: {
     dataBound: function () {
         var spreadsheet = this.$refs.spreadsheet;
         if (spreadsheet.ej2Instances.activeSheetIndex === 0) {
            spreadsheet.sort({sortDescriptors: { field: 'F',  sortComparer:
this.mySortComparer }, containsHeader: true}, 'A1:H20');
         }
     },
     sortComplete: function (args) {
         var spreadsheet = this.$refs.spreadsheet;
         spreadsheet.selectRange(args.range);
         // code here.
     },
     mySortComparer: function (x, y) {
    // custom sort comparer to sort based on the custom list.
     var customList = ['Perfect', 'Sufficient', 'Insufficient'];
     var comparer = DataUtil.fnSort('Ascending');
     return comparer(x ? customList.indexOf(x.value) : x, y ?
customList.indexOf(y.value) : y);
      };
   }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/custom-sort-cs1" %}

See Also

- Filtering
- Sorting
- Hyperlink

Print in Vue Spreadsheet component

You can use the print method by importing from ej2-base package. Here, the Select event in the dropdown and the dataBound event are used to print the single/multiple sheets of data. To print the

single/multiple sheets, use the dropdown button and select the Print (or) Print All option. In the following sample, you can be able to print the single/multiple sheets.

APP.VUE

```
<template><div>
<ejs-dropdownbutton :items='items' :select='itemSelect'>Print</ejs-
dropdownbutton>
  <ejs-spreadsheet id="spreadsheet" ref="spreadsheet" :created="created"
:dataBound="dataBound">
                <e-sheets>
                  <e-sheet name="Budget">
                    <e-ranges>
                      <e-range :dataSource="dataSource1"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                    </e-columns>
                  </e-sheet>
                  <e-sheet name="Salary">
                    <e-ranges>
                      <e-range :dataSource="dataSource2"></e-range>
                    </e-ranges>
                    <e-columns>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      <e-column :width=100></e-column>
                      </e-columns>
                  </e-sheet>
                </e-sheets>
              </ejs-spreadsheet></div>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { salaryData, budgetData  } from './data.js';
import { DropDownButtonPlugin } from "@syncfusion/ej2-vue-splitbuttons";
import { createElement, getComponent, print } from '@syncfusion/ej2-base';
Vue.use(SpreadsheetPlugin);
Vue.use(DropDownButtonPlugin);
var printElement = createElement("div", {
  className: "e-sheet-panel",
  innerHTML:
    '<div class="e-spreadsheet-print"></div><div class="e-sheet"><div
class="e-main-panel style="height:100%" style="overflow: unset"><div
class="e-sheet-content" ></div></div></div>'
}) // creating same hierarchy of element as DOM
var isPrint = false;
export default {
   data: () => {
    return {
      dataSource1: budgetData,
      dataSource2: salaryData,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      items:[
        {
          text: "Print"
        },
        {
          text: "Print All"
        }]
    }
  },
  methods: {
   created: function () {
       var spreadsheet = this.$refs.spreadsheet;
     spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:D1');
        spreadsheet.cellFormat({ fontWeight: 'bold'}, 'A11:D11');
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'Salary!A1:D1');
      },
      itemSelect: function(args) {
      var spreadsheet = getComponent(document.getElementById("spreadsheet"),
"spreadsheet");
      if (args.item.text === 'Print') {
      printElement.querySelector(".e-sheet-content").innerHTML =
document.querySelector(
        ".e-sheet-content"
      ).outerHTML; //  To add the spreadsheet table
      var usedRange = spreadsheet.getActiveSheet().usedRange;
      var tbody = printElement.querySelector('tbody');
      for (var i = tbody.getElementsByClassName('e-row').length; i >= 0; i--
) {
        if (tbody.getElementsByClassName('e-row')[i] &&
parseInt(tbody.getElementsByClassName('e-row')[i].getAttribute('aria-
rowindex'), 10) > usedRange.rowIndex + 1) {
          tbody.getElementsByClassName('e-row')[i].remove();
        }
      }
      (printElement.querySelector('.e-sheet-
content').children[0].getElementsByClassName('e-
virtualtrack')[0]).style.height = 'auto';
      print(printElement);
      printElement.querySelector(".e-sheet-content").innerHTML = '';
    }
    if (args.item.text === 'Print All') {
      var sheets = spreadsheet.sheets;
      if (spreadsheet.activeSheetIndex === 0) {
        printElement.querySelector(".e-sheet-content").innerHTML =
document.querySelector(
          ".e-sheet-content"
        ).outerHTML; //  To add the spreadsheet table
        var usedRange1 = spreadsheet.getActiveSheet().usedRange;
        var tbody1 = printElement.querySelector('tbody');
        for (var a = tbody1.getElementsByClassName('e-row').length; a >= 0;
a--) {
          if (tbody1.getElementsByClassName('e-row')[a] &&
parseInt(tbody1.getElementsByClassName('e-row')[a].getAttribute('aria-
rowindex'), 10) > usedRange1.rowIndex + 1) {
            tbody1.getElementsByClassName('e-row')[a].remove();
```

```
            }
          }
          if (sheets[spreadsheet.activeSheetIndex + 1]) {
            spreadsheet.goTo(sheets[spreadsheet.activeSheetIndex + 1].name +
"!A1");
            isPrint = true;
          } else {
            print(printElement);
            printElement.querySelector(".e-sheet-content").innerHTML = '';
          }
        } else {
          if (sheets[0]) {
            spreadsheet.goTo(sheets[0].name + "!A1");
            isPrint = true;
          }
        }
      }
    },
    dataBound() {
        var spreadsheet =
getComponent(document.getElementById("spreadsheet"), "spreadsheet");
      if (isPrint) {
        printElement.querySelector(
          '.e-sheet-content'
        ).innerHTML += document.querySelector('.e-sheet-content').outerHTML;
        var usedRange = spreadsheet.getActiveSheet()
          .usedRange;
        var tbody = printElement
          .querySelector('.e-sheet-content')
          .children[spreadsheet.activeSheetIndex].querySelector('tbody');
        for (
          var i = tbody.getElementsByClassName('e-row').length;
          i >= 0;
          i--
        ) {
          if (
            tbody.getElementsByClassName('e-row')[i] &&
            parseInt(tbody.getElementsByClassName('e-
row')[i].getAttribute('aria-rowindex'), 10) > usedRange.rowIndex + 1) {
            tbody.getElementsByClassName('e-row')[i].remove();
          }
        }
        var sheets = spreadsheet.sheets;
        if (sheets.length - 1 === spreadsheet.activeSheetIndex) {
          var count = printElement.querySelector('.e-sheet-content')
            .childElementCount;
          if (count > 1) {
            for (var j = 0; j < count; j++) {
              (printElement
                .querySelector('.e-sheet-content')
                .children[j].getElementsByClassName(
                  'e-virtualtrack'
                )[0]).style.height = 'auto';
              printElement
                .querySelector('.e-sheet-content')
                .children[j].setAttribute('style', 'page-break-after:
always;');
```

```
            }
          }
          print(printElement);
          isPrint = false;
          printElement.querySelector('.e-sheet-content').innerHTML = '';
        } else {
          if (sheets[spreadsheet.activeSheetIndex + 1]) {
            spreadsheet.goTo(
              sheets[spreadsheet.activeSheetIndex + 1].name + '!A1'
            );
          }
        }
      }
    }
      }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
 @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
 @import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
 @import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/print-cs1" %}

Changing the active sheet while importing a file in Vue Spreadsheet component

You can change the active sheet of imported file by updating activeSheetIndex property on the openComplete event.

The following code example shows how to set the active sheet when importing an Excel file.

APP.VUE

```
<template>
  <ejs-spreadsheet ref="spreadsheet" :openUrl="openUrl" :saveUrl="saveUrl"
:openComplete="openComplete"></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
Vue.use(SpreadsheetPlugin);
export default {
  data: () => {
    return {
      openUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/open',
      saveUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/save'
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
  methods: {
    openComplete: function () {
      var spreadsheet = this.$refs.spreadsheet;
      if (spreadsheet) {
        spreadsheet.activeSheetIndex = 2;
      }
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/change-active-sheet-cs1" %}

Import an excel document using file uploader in ##Platform_Name## Spreadsheet component

If you explore your machine to select and upload an excel document using the file uploader, you will receive the uploaded document as a raw file in the success event of the file uploader. In this success event, you should pass the received raw file as an argument to the Spreadsheet's open method to see the appropriate output.

The following code example shows how to import an excel document using file uploader in spreadsheet.

APP.VUE

```
<template>
  <div>
    <ejs-spreadsheet
      id="spreadsheet"
      ref="spreadsheet"
      :openUrl="openUrl"
    ></ejs-spreadsheet>
    <ejs-uploader
      :asyncSettings="path"
      :success="onSuccess"
      :allowedExtensions="allowExtentions"
    ></ejs-uploader>
  </div>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { UploaderPlugin } from "@syncfusion/ej2-vue-inputs";
import { getComponent } from "@syncfusion/ej2-base";
```

```
Vue.use(UploaderPlugin);
Vue.use(SpreadsheetPlugin);
export default {
  data: () => {
    return {
      allowExtentions: ".xlsx, .xls, .csv",
      path: {
        saveUrl:

"https://services.syncfusion.com/vue/production/api/FileUploader/Save",
        removeUrl:

"https://services.syncfusion.com/vue/production/api/FileUploader/Remove",
      },
      openUrl:

"https://services.syncfusion.com/vue/production/api/spreadsheet/open",
    };
  },
  methods: {
    onSuccess: (args) => {
      let spreadsheet = getComponent(
        document.getElementById("spreadsheet"),
        "spreadsheet"
      );
      if (args.operation === "upload")
        spreadsheet.open({ file: args.file.rawFile });
    },
  },
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/open-uploader-cs1" %}

Identify the context menu opened in Vue Spreadsheet component

The Spreadsheet includes several context menus that will open and display depending on the action. When you right-click on a cell, for example, a context menu with options related to the cell element appears.

The class name returned by the contextMenuBeforeOpen event can be used to identify the context menu that is opened. The context menus and their class names are tabulated below.

| Class name | Context menu name |

|-------|---------|

| .e-sheet-content | Cell context menu |

| .e-toolbar-item | Footer context menu |

| .e-rowhdr-table | Row header context menu |

| .e-colhdr-table | Column header context menu |

The following code example shows how to identify the context menu opened.

APP.VUE

```
<template>
  <ejs-spreadsheet
    ref="spreadsheet"
    :contextMenuBeforeOpen="contextMenuBeforeOpen"
  ></ejs-spreadsheet>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { closest } from "@syncfusion/ej2-base";
Vue.use(SpreadsheetPlugin);
export default {
  methods: {
    contextMenuBeforeOpen: function (args) {
      if (closest(args.event.target, ".e-sheet-content")) {
        console.log("Cell Context Menu");
      } else if (closest(args.event.target, ".e-colhdr-table")) {
        console.log("Column Header Context Menu");
      } else if (closest(args.event.target, ".e-rowhdr-table")) {
        console.log("Row Header Context Menu");
      } else if (closest(args.event.target, ".e-toolbar-item")) {
        console.log("Footer Context Menu");
      }
    },
  },
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/find-target-context-menu" %}

Copyright © 2001 -2024 Syncfusion Inc.

Save and open Spreadsheet data as a Base64 string in Vue Spreadsheet component

In the Spreadsheet component, there is currently no direct option to save and open data as a Base64 string. You can achieve this by saving the Spreadsheet data as blob data and then converting that saved blob data to a Base64 string using FileReader.

You can get the Spreadsheet data as blob in the saveComplete event when you set the needBlobData as **true** and isFullPost as **false** in the beforeSave event.

The following code example shows how to save and open the spreadsheet data as base64 string.

APP.VUE

```
<template>
  <div>
    <ejs-button class="e-btn custom-btn" v-
on:click.native="importBtn">Import Base64</ejs-button>
    <ejs-button class="e-btn custom-btn" v-
on:click.native="exportBtn">Export as Base64</ejs-button>
    <ejs-spreadsheet ref="spreadsheet" :openUrl="openUrl"
:beforeSave="beforeSave" :saveComplete="saveComplete">
      <e-sheets>
        <e-sheet name="Car Sales Report">
          <e-ranges>
            <e-range :dataSource="dataSource"></e-range>
          </e-ranges>
          <e-columns>
            <e-column :width=180></e-column>
            <e-column :width=130></e-column>
            <e-column :width=130></e-column>
            <e-column :width=180></e-column>
            <e-column :width=120></e-column>
            <e-column :width=130></e-column>
          </e-columns>
        </e-sheet>
      </e-sheets>
    </ejs-spreadsheet>
  </div>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data } from './data.js';
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(SpreadsheetPlugin);
var base64String;
export default {
  data: () => {
    return {
      dataSource: data,
      openUrl:
'https://services.syncfusion.com/vue/production/api/spreadsheet/open',
    }
  },
  methods: {
    beforeSave: function (args) {
```

```
      args.needBlobData = true; // To trigger the saveComplete event.
      args.isFullPost = false; // Get the spreadsheet data as blob data in
the saveComplete event.
    },
    saveComplete: function (args) {
      // Convert blob data to base64 string.
      var reader = new FileReader();
      reader.readAsDataURL(args.blobData);
      reader.onloadend = function () {
        base64String = reader.result;
      };
    },
    importBtn: function () {
      var spreadsheet = this.$refs.spreadsheet;
      fetch(base64String)
        .then((response) => response.blob())
        .then((fileBlob) => {
          var file = new File([fileBlob], 'Sample.xlsx');
          spreadsheet.open({ file: file });
        });
    },
    exportBtn: function () {
      var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.save({
        url:
'https://services.syncfusion.com/vue/production/api/spreadsheet/save',
        fileName: 'Worksheet',
        saveType: 'Xlsx',
      }); // Specifies the save URL, file name, file type need to be saved.
      // Logs base64 string into the console.
      console.log('Base64 String - ', base64String);
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
.custom-btn {
  margin-bottom: 10px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/base-64-string" %}

Copyright © 2001 -2024 Syncfusion Inc.

Insert a sheet programmatically and make it the active sheet in Vue Spreadsheet component

A sheet is a collection of cells organized in the form of rows and columns that allows you to store, format, and manipulate the data. Using insertSheet method, you can insert one or more sheets at the desired index. Then, you can make the inserted sheet as active sheet by focusing the start cell of that sheet using the goTo method.

The following code example shows how to insert a sheet programmatically and make it the active sheet.

APP.VUE

```
<template>
  <div>
    <ejs-button class="e-btn custom-btn" v-
on:click.native="insertSheet">Insert Sheet</ejs-button>
    <ejs-spreadsheet ref="spreadsheet">
      <e-sheets>
        <e-sheet name="Car Sales Report">
          <e-ranges>
            <e-range :dataSource="dataSource"></e-range>
          </e-ranges>
          <e-columns>
            <e-column :width=180></e-column>
            <e-column :width=130></e-column>
            <e-column :width=130></e-column>
            <e-column :width=180></e-column>
            <e-column :width=130></e-column>
            <e-column :width=120></e-column>
          </e-columns>
        </e-sheet>
      </e-sheets>
    </ejs-spreadsheet>
  </div>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { data, employeeData } from './data.js';
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ButtonPlugin);
Vue.use(SpreadsheetPlugin);
export default {
  data: () => {
    return {
      dataSource: data,
    }
  },
  methods: {
    insertSheet: function () {
      var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.insertSheet(
        [
          {
            index: 1,
            name: 'new_sheet',
            ranges: [
              {
                dataSource: employeeData,
```

```
                startCell: 'A1'
              },
            ]
          },
        ]
      );
      // Use the timeout function to wait until the sheet is inserted.
      setTimeout(() => {
        // Method for switching to a new sheet.
        spreadsheet.goTo('new_sheet!A1');
      })
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
.custom-btn {
  margin-bottom: 10px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/insert-sheet-change-active-sheet-cs1" %}

Get the filtered rows in Vue Spreadsheet component

Filtering allows you to view specific rows in a spreadsheet while hiding the others. The allowFiltering property allows you to enable or disable filtering functionality through the UI. You can also use the allowFiltering property and applyFilter method combination to filter data via code behind. The filtered rows can be identified by iterating through the row collection on the sheet and using the isFiltered property available in each row object.

The following code example shows how to get the filtered rows.

APP.VUE

```
<template>
  <div>
    <ejs-button class="e-btn custom-btn" @click="getFilterData">Get Filtered
Data</ejs-button>
    <ejs-spreadsheet ref="spreadsheet" :created="created">
      <e-sheets>
        <e-sheet>
          <e-ranges>
            <e-range :dataSource="dataSource"></e-range>
          </e-ranges>
```

```
          <e-columns>
            <e-column :width="width1"></e-column>
            <e-column :width="width2"></e-column>
            <e-column :width="width2"></e-column>
            <e-column :width="width1"></e-column>
            <e-column :width="width2"></e-column>
            <e-column :width="width3"></e-column>
          </e-columns>
        </e-sheet>
      </e-sheets></ejs-spreadsheet>
  </div>
</template>
<script>
import Vue from "vue";
import { SpreadsheetPlugin } from "@syncfusion/ej2-vue-spreadsheet";
import { defaultData } from './data.js';
Vue.use(SpreadsheetPlugin);
export default {
  data: () => {
    return {
      dataSource: defaultData,
      width1: 180,
      width2: 130,
      width3: 120,
    }
  },
  methods: {
    created: function () {
      var spreadsheet = this.$refs.spreadsheet;
      spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center',
verticalAlign: 'middle' }, 'A1:F1');
      var predicates = [{
        field: 'C',
        operator: 'equal',
        value: 'Pink',
        matchCase: false
      }];
      spreadsheet.applyFilter(predicates, 'A1:C7');
    },
    getFilterData: function () {
      var spreadsheet = this.$refs.spreadsheet;
      var activeSheet = spreadsheet.ej2Instances.getActiveSheet();
      var usedRange = activeSheet.usedRange;
      for (var i = 0; i <= usedRange.rowIndex; i++) {
        // Get the filtered row using isFiltered property.
        var filteredRow = (activeSheet.rows[i]).isFiltered;
        if (!filteredRow) {
          var rowData = spreadsheet.getRowData(i);
          console.log("Row:", i + 1, "Cells", rowData);
        }
      }
    },
  }
}
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-vue-
spreadsheet/styles/material.css";
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
@import "../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css";
.custom-btn {
  margin: 0 0 10px 0;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/spreadsheet/filter-cs2" %}

Mobile responsiveness in Vue Spreadsheet component

The Spreadsheet control rendered in desktop mode will be adaptive in all mobile devices where the layout gets adjusted based on their parent element’s dimensions to accommodate any resolution.

You can see the overflowed items of ribbon header, ribbon content, and sheet tab using touch and swipe action. The right navigation arrow is added at the end of the ribbon content through which the user can navigate towards overflowed items. Once you reached the rightmost end of the ribbon content, the right navigation arrow will change to left navigation arrow through which you can navigate to the left of the ribbon content.

Copyright © 2001 -2024 Syncfusion Inc.

Feature list in Vue Spreadsheet component

The following table compares Excel functionality with the availability of EJ1 and EJ2 Spreadsheet features.

| **Features** | **Available in EJ1 Spreadsheet** | **Available in EJ2 Spreadsheet** | **Comments** |

| --- | --- | --- | --- |

| Ribbon | Yes | Yes | - |

| Formula bar | Yes | Yes | - |

| Sheet tab | Yes | Yes | - |

| Show / Hide gridlines and header | Yes | Yes | - |

| Scrolling | Partially | Yes | - |

| Selection | Yes | Yes | - |

| Editing | Yes | Yes | - |

Copyright © 2001 -2024 Syncfusion Inc.

Formulae	Yes	Partially	EJ2 supports limited number of most used formulas
Named range	Yes	Partially	EJ2 Spreadsheet Named range supports only in workbook scope
Data Binding	Yes	Yes	-
Formatting	Yes	Yes	-
Context menu	Yes	Yes	-
Keyboard navigation	Yes	Yes	-
Keyboard shortcuts	Yes	Yes	-
Sorting	Yes	Yes	-
Filtering	Yes	Yes	-
Hyperlink	Yes	Yes	-
Undo & redo	Yes	Yes	-
Open and Save	Yes	Yes	-
Resize / Autofit	Yes	Yes	-
Clipboard	Yes	Yes	-
Collaborative editing	No	Yes	-
Wrap text	Yes	Yes	-
Template	No	Yes	-
Merge cells	Yes	Yes	-
Show / Hide rows and columns	Yes	Yes	-
Sheet customizations	Yes	Partially	Move or copy sheet is not supported in EJ2 spreadsheet.
Data Validation	Yes	Yes	-
Table	Yes	No	-
Chart	Yes	Yes	-
Image	Yes	Yes	-
Conditional formatting	Yes	Yes	-
Freeze Pane	Yes	Yes	-
Scaling	No	No	-
Print	Yes	No	-
Grouping	No	No	-
Autofill	Yes	No	-
Auto Sum	Yes	Yes	-
Format painter	Yes	No	-
Cell Style	Yes	Partially	We can only customize the cell style in EJ2 Spreadsheet through API.

Copyright © 2001 -2024 Syncfusion Inc.

| Protection | Yes | Partially | Custom encryption is not supported in EJ2 Spreadsheet's protect workbook. |

| Find and replace | Yes | Yes | - |

| Drag and Drop | Yes | No | - |

| Notes | Yes | No | - |

| Comments | No | No | - |

| Pivot table | Yes | No | - |

| Sparklines | Yes | No | - |

| Form controls | Yes | No | - |

| Shapes | No | No | - |

| Clear | Yes | Yes | - |

See Also

- Cell range
- Formatting
- Hyperlink
- Undo and Redo
- Unlock the particular cells in the protected sheet

Stepper

Getting Started with the Vue Stepper Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Stepper component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies are required to use the Stepper component in your application.

`js

|-- @syncfusion/ej2-vue-navigations

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-popups

|-- @syncfusion/ej2-navigations

|-- @syncfusion/ej2-vue-base

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

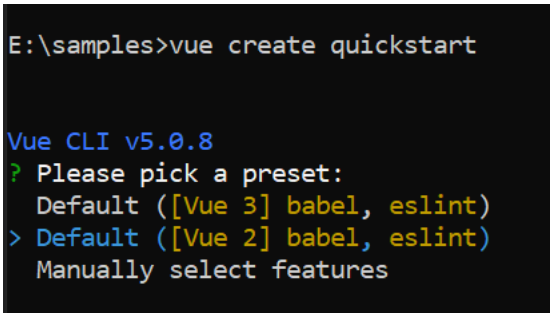
Copyright © 2001 -2024 Syncfusion Inc.

`bash
npm install -g @vue/cli
vue create quickstart
cd quickstart
npm run serve
`

or

`bash
yarn global add @vue/cli
vue create quickstart
cd quickstart
yarn run serve
`

When creating a new project, choose the option `Default ([Vue 2] babel, eslint)` from the menu.



Once the `quickstart` project is set up with default settings, proceed to add Syncfusion components to the project.

Adding Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

Install the `@syncfusion/ej2-vue-navigations` package by running the following command:

`bash
npm install @syncfusion/ej2-vue-navigations --save
`

or

`bash
yarn add @syncfusion/ej2-vue-navigations
`

Copyright © 2001 -2024 Syncfusion Inc.

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Stepper component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
</style>
```

Adding Syncfusion Vue component

Follow the below steps to add the Vue Stepper component using Composition API or Options API:

1\. First, import and register the Stepper component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { StepperComponent as EjsStepper } from "@syncfusion/ej2-vue-
navigations";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { StepperComponent } from "@syncfusion/ej2-vue-navigations";
export default {
components: {
'ejs-stepper': StepperComponent
}
}
</script>
```

2\. In the template section, define the Stepper component.

~/SRC/APP.VUE

```
<template>
<ejs-stepper id="stepper"></ejs-stepper>
</template>
```

Adding steps

You can use the <e-step> directive within the <e-steps> directive to define each step.

Copyright © 2001 -2024 Syncfusion Inc.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-stepper id="stepper">
<e-steps>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
</e-steps>
</ejs-stepper>
</template>
<script setup>
import { StepperComponent as EjsStepper, StepsDirective as ESteps,
StepDirective as EStep  } from "@syncfusion/ej2-vue-navigations";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-stepper id="stepper">
<e-steps>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
</e-steps>
</ejs-stepper>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
components: {
'ejs-stepper': StepperComponent,
"e-steps": StepsDirective,
"e-step": StepDirective
}
};
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <ejs-stepper id="stepper">
      <e-steps>
        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
      </e-steps>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  </ejs-stepper>
</template>
<script setup>
  import { StepperComponent as EjsStepper, StepsDirective as ESteps,
StepDirective as EStep  } from "@syncfusion/ej2-vue-navigations";
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <ejs-stepper id="stepper">
        <e-steps>
          <e-step></e-step>
          <e-step></e-step>
          <e-step></e-step>
          <e-step></e-step>
          <e-step></e-step>
        </e-steps>
  </ejs-stepper>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
    components: {
      'ejs-stepper': StepperComponent,
      "e-steps": StepsDirective,
      "e-step": StepDirective
    }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

Copyright © 2001 -2024 Syncfusion Inc.

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/stepper/getting-started-cs1" %}

Configure icon and label

You can define the step icon and label by setting the iconCss and label properties using <e-step> directive.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div class="iconWithLabel">
    <ejs-stepper id="stepper">
        <e-steps>
            <e-step label='Cart' iconCss='sf-icon-cart'></e-step>
            <e-step label='Address' iconCss='sf-icon-user'></e-step>
            <e-step label='Delivery' iconCss='sf-icon-transport'></e-step>
            <e-step label='Payment' iconCss='sf-icon-payment'></e-step>
            <e-step label='Ordered' iconCss='sf-icon-success'></e-step>
        </e-steps>
    </ejs-stepper>
  </div>
</template>

<script setup>
  import { StepperComponent as EjsStepper, StepsDirective as ESteps,
StepDirective as EStep  } from "@syncfusion/ej2-vue-navigations";
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
PPPUACwQAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
```

```
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  .iconWithLabel [class^="sf-icon-"], .iconWithLabel [class*=" sf-icon-"] {
      font-family: 'Default' !important;
      speak: none;
      font-style: normal;
      font-weight: normal;
      font-variant: normal;
```

```
      text-transform: none;
      line-height: inherit;
      -webkit-font-smoothing: antialiased;
      -moz-osx-font-smoothing: grayscale;
  }
  .iconWithLabel .sf-icon-cart:before { content: "\e710"; }
  .iconWithLabel .sf-icon-user:before { content: "\e708"; }
  .iconWithLabel .sf-icon-transport:before { content: "\e702"; }
  .iconWithLabel .sf-icon-payment:before { content: "\e706"; }
  .iconWithLabel .sf-icon-success:before { content: "\e715"; }

</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div class="iconWithLabel">
    <ejs-stepper id="stepper">
        <e-steps>
          <e-step label='Cart' iconCss='sf-icon-cart'></e-step>
          <e-step label='Address' iconCss='sf-icon-user'></e-step>
          <e-step label='Delivery' iconCss='sf-icon-transport'></e-step>
          <e-step label='Payment' iconCss='sf-icon-payment'></e-step>
          <e-step label='Ordered' iconCss='sf-icon-success'></e-step>
        </e-steps>
  </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
    components: {
      'ejs-stepper': StepperComponent,
      "e-steps": StepsDirective,
      "e-step": StepDirective
    }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
```

```
    }
    .iconWithLabel [class^="sf-icon-"], .iconWithLabel [class*=" sf-icon-"] {
        font-family: 'Default' !important;
        speak: none;
        font-style: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
        line-height: inherit;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
    }
    .iconWithLabel .sf-icon-cart:before { content: "\e710"; }
    .iconWithLabel .sf-icon-user:before { content: "\e708"; }
    .iconWithLabel .sf-icon-transport:before { content: "\e702"; }
    .iconWithLabel .sf-icon-payment:before { content: "\e706"; }
    .iconWithLabel .sf-icon-success:before { content: "\e715"; }

</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/iconWithLabel" %}

Getting Started with Syncfusion Stepper Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Stepper component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

Copyright © 2001 -2024 Syncfusion Inc.

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

To use the Vue Stepper component in the project, the @syncfusion/ej2-vue-navigations package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-navigations --save

`

or

`bash

yarn add @syncfusion/ej2-vue-navigations

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Button component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Adding Syncfusion Vue Stepper component

Follow the below steps to add the Vue Stepper component using Composition API or Options API:

Importing the component and directives

First, import and register the Stepper component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { StepperComponent as EjsStepper, StepsDirective as ESteps,
StepDirective as EStep  } from "@syncfusion/ej2-vue-navigations";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
components: {
'ejs-stepper': StepperComponent,
"e-steps": StepsDirective,
"e-step": StepDirective
}
};
</script>
```

Adding the Stepper component

Add the component definition in template section.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-stepper id="stepper"></ejs-stepper>
</template>
<script setup>
import { StepperComponent as EjsStepper } from "@syncfusion/ej2-vue-
navigations";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-stepper id="stepper"></ejs-stepper>
</template>
<script>
import { StepperComponent } from "@syncfusion/ej2-vue-navigations";
export default {
components: {
'ejs-stepper': StepperComponent
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Adding steps

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-stepper id="stepper">
<e-steps>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
</e-steps>
</ejs-stepper>
</template>
<script setup>
import { StepperComponent as EjsStepper, StepsDirective as ESteps,
StepDirective as EStep  } from "@syncfusion/ej2-vue-navigations";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-stepper id="stepper">
<e-steps>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
</e-steps>
</ejs-stepper>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
components: {
'ejs-stepper': StepperComponent,
"e-steps": StepsDirective,
"e-step": StepDirective
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class="iconWithLabel">
<ejs-stepper id="stepper">
<e-steps>
<e-step label='Cart' iconCss='sf-icon-cart'></e-step>
<e-step label='Address' iconCss='sf-icon-user'></e-step>
<e-step label='Delivery' iconCss='sf-icon-transport'></e-step>
<e-step label='Payment' iconCss='sf-icon-payment'></e-step>
<e-step label='Ordered' iconCss='sf-icon-success'></e-step>
</e-steps>
</ejs-stepper>
</div>
</template>
<script setup>
import { StepperComponent as EjsStepper, StepsDirective as ESteps,
StepDirective as EStep  } from "@syncfusion/ej2-vue-navigations";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@font-face {
font-family: 'Default';
src:
url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAA
```

```
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
font-weight: normal;
font-style: normal;
}
.iconWithLabel [class^="sf-icon-"], .iconWithLabel [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
```

```
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.iconWithLabel .sf-icon-cart:before { content: "\e710"; }
.iconWithLabel .sf-icon-user:before { content: "\e708"; }
.iconWithLabel .sf-icon-transport:before { content: "\e702"; }
.iconWithLabel .sf-icon-payment:before { content: "\e706"; }
.iconWithLabel .sf-icon-success:before { content: "\e715"; }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="iconWithLabel">
<ejs-stepper id="stepper">
<e-steps>
<e-step label='Cart' iconCss='sf-icon-cart'></e-step>
<e-step label='Address' iconCss='sf-icon-user'></e-step>
<e-step label='Delivery' iconCss='sf-icon-transport'></e-step>
<e-step label='Payment' iconCss='sf-icon-payment'></e-step>
<e-step label='Ordered' iconCss='sf-icon-success'></e-step>
</e-steps>
</ejs-stepper>
</div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
components: {
'ejs-stepper': StepperComponent,
"e-steps": StepsDirective,
"e-step": StepDirective
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@font-face {
font-family: 'Default';
src:
url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxAQExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
font-weight: normal;
font-style: normal;
```

```
}
.iconWithLabel [class^="sf-icon-"], .iconWithLabel [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.iconWithLabel .sf-icon-cart:before { content: "\e710"; }
.iconWithLabel .sf-icon-user:before { content: "\e708"; }
.iconWithLabel .sf-icon-transport:before { content: "\e702"; }
.iconWithLabel .sf-icon-payment:before { content: "\e706"; }
.iconWithLabel .sf-icon-success:before { content: "\e715"; }
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Steps in Vue Stepper component

The Vue Stepper allows you to add steps using the <e-step> directive . Each step can be configured with options such as iconCss, text, label, cssClass and more.

Adding steps

You can define the icon and text content for each step using the iconCss, text and label properties.

Copyright © 2001 -2024 Syncfusion Inc.

Defining icon CSS

You can define the CSS class to show the icon for each step using the iconCss property.

APP.VUE

```
<template>
  <div class="step-icon">
    <ejs-stepper id="stepper">
      <e-steps>
        <e-step iconCss='sf-icon-cart'></e-step>
        <e-step iconCss='sf-icon-transport'></e-step>
        <e-step iconCss='sf-icon-payment'></e-step>
        <e-step iconCss='sf-icon-success'></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
```

```
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  .sf-icon-cart:before { content: "\e710"; }
  .sf-icon-transport:before { content: "\e702"; }
  .sf-icon-payment:before { content: "\e706"; }
  .sf-icon-success:before { content: "\e715"; }
  .step-icon {
    margin-top: 30px;
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/icon" %}

Defining text content

You can define text instead of an icon by setting the `text` property and display label content for a step using the `label` property.

When both label and text are defined, the label takes priority for display based on the `stepType`.

APP.VUE

```
<template>
  <div class="step-text">
    <ejs-stepper id="stepper" stepType="indicator">
      <e-steps>
        <e-step text='A'></e-step>
        <e-step text='B'></e-step>
        <e-step text='C'></e-step>
        <e-step text='D'></e-step>
      </e-steps>
    </ejs-stepper>
    <ejs-stepper id="labelStepper">
      <e-steps>
        <e-step label='Cart'></e-step>
        <e-step label='Delivery Address'></e-step>
        <e-step label='Payment'></e-step>
        <e-step label='Confirmation'></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  .step-text {
    margin-top: 30px;
    padding: 30px;
  }
  #labelStepper {
    margin-top: 50px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/text" %}

Optional steps

You can show whether the step is optional or not by using optional property. By default, the optional property is false.

APP.VUE

```
<template>
  <div class="optional-step">
    <ejs-stepper id="stepper">
      <e-steps>
        <e-step iconCss='sf-icon-cart'></e-step>
        <e-step iconCss='sf-icon-transport'></e-step>
        <e-step iconCss='sf-icon-payment' optional=true></e-step>
        <e-step iconCss='sf-icon-success'></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
```

```
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IEReSFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAA
```

```
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  .sf-icon-cart:before { content: "\e710"; }
  .sf-icon-transport:before { content: "\e702"; }
  .sf-icon-payment:before { content: "\e706"; }
  .sf-icon-success:before { content: "\e715"; }
  .optional-step {
    margin-top: 30px;
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/optional" %}

Disabling steps

You can use the disabled property to disable a step, preventing user interaction when set to true. By default, the value is false.

APP.VUE

```
<template>
  <div class="disabled-step">
    <ejs-stepper id="stepper">
      <e-steps>
        <e-step iconCss='sf-icon-cart'></e-step>
        <e-step iconCss='sf-icon-transport'></e-step>
        <e-step iconCss='sf-icon-payment' disabled="true"></e-step>
        <e-step iconCss='sf-icon-success'></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  }
};
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
```

```
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRyby BTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdABhAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  .sf-icon-cart:before { content: "\e710"; }
  .sf-icon-transport:before { content: "\e702"; }
  .sf-icon-payment:before { content: "\e706"; }
  .sf-icon-success:before { content: "\e715"; }
  .disabled-step {
    margin-top: 30px;
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/disabled" %}

Setting active step

You can set the active step by specifying its index using the activeStep property. The default value is 0.

APP.VUE

```
<template>
  <div class="step-icon">
    <ejs-stepper id="stepper" :activeStep=1>
      <e-steps>
        <e-step iconCss='sf-icon-cart'></e-step>
        <e-step iconCss='sf-icon-transport'></e-step>
        <e-step iconCss='sf-icon-payment'></e-step>
        <e-step iconCss='sf-icon-success'></e-step>
      </e-steps>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
```

```
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxAQExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRyby BTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdABhAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  .sf-icon-cart:before { content: "\e710"; }
  .sf-icon-transport:before { content: "\e702"; }
  .sf-icon-payment:before { content: "\e706"; }
  .sf-icon-success:before { content: "\e715"; }
  .step-icon {
    margin-top: 30px;
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/activeStep" %}

Copyright © 2001 -2024 Syncfusion Inc.

Step status

Each step's progress state can be specified using the `status` property. The possible values are `NotStarted`, `InProgress` and `Completed`. By default, the value is `NotStarted`.

APP.VUE

```
<template>
  <div class="stepper-section">
    <ejs-stepper id="stepper" ref="stepper" :stepChanged="stepChanged">
      <e-steps>
        <e-step label='Cart' iconCss='sf-icon-cart'></e-step>
        <e-step label='Payment' iconCss='sf-icon-payment'></e-step>
        <e-step label='Ordered' iconCss='sf-icon-success'></e-step>
      </e-steps>
    </ejs-stepper>
    <div id="paymentStatus" style="margin-top: 50px">Your payment has not
started yet</div>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  },
  methods: {
    stepChanged: function() {
      this.updateStatus(this.$refs.stepper.ej2Instances.steps[1].status);
    },
    updateStatus: function (stepStatus) {
      let statusMap = {
        'NotStarted': { text: 'Your payment has not started yet', color:
'#e74d4d' },
        'InProgress': { text: 'Processing your payment', color: 'orange' },
        'Completed': { text: 'Payment successful', color: '#4CAF50' }
      };
      let currentStatus = document.getElementById("paymentStatus");
      if (currentStatus) {
          let { text, color } = statusMap[stepStatus];
          currentStatus.innerText = text;
          currentStatus.style.backgroundColor = color;
      }
    }
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  #paymentStatus {
    padding: 10px;
    background-color: #e74d4d;
    color: white;
    border-radius: 5px;
    margin: 10px auto;
    text-align: center;
    font-weight: bold;
    max-width: 350px;
}
@font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgQEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
```

```
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxAQExEQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRyby BTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdABhAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }
.stepper-section {
    width: 75%;
    margin: 0px auto;
    min-width: 85px;
    padding: 25px 0;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/stepStatus" %}

Step styling

You can use the cssClass property to customize the appearance of each step.

APP.VUE

```
<template>
  <div class="stepper-customize">
    <ejs-stepper id="iconStepper">
      <e-steps>
        <e-step iconCss='sf-icon-cart' label='Cart'></e-step>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <e-step iconCss='sf-icon-transport' label='Delivery Address'></e-
step>
        <e-step iconCss='sf-icon-payment' label='Payment' cssClass="custom-
step" optional=true></e-step>
        <e-step iconCss='sf-icon-success' label='Confirmation'></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
```

```
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRyby8TdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  .sf-icon-cart:before { content: "\e710"; }
  .sf-icon-transport:before { content: "\e702"; }
  .sf-icon-payment:before { content: "\e706"; }
  .sf-icon-success:before { content: "\e715"; }
  #iconStepper .custom-step .e-step-label-optional {
    font-style: italic;
    font-weight: 900;
    color: #299100;
```

```
  }
  .stepper-customize {
    margin-top: 30px;
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/cssclass" %}

Step validation

You can set the validation state for each step to displaying a success or error icon by using isValid property.

To know more about Stepper validation, please refer to the Validation section.

Step types in Vue Stepper component

The Stepper component provides support for displaying steps with the following step types.

Default type

In default type, the Stepper displays steps with a combination of both indicators and labels by setting the stepType property as Default. By default, the Stepper displays steps in the Default type.

APP.VUE

```
<template>
  <div class="step-default">
    <ejs-stepper id="stepper" stepType="default">
        <e-steps>
          <e-step label='Cart' iconCss='sf-icon-cart'></e-step>
          <e-step label='Delivery' iconCss='sf-icon-transport'></e-step>
          <e-step label='Payment' iconCss='sf-icon-payment'></e-step>
          <e-step label='Ordered' iconCss='sf-icon-success'></e-step>
        </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
    components: {
      'ejs-stepper': StepperComponent,
      "e-steps": StepsDirective,
      "e-step": StepDirective
    }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @font-face {
    font-family: 'Default';
```

```
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
```

```
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  .step-default [class^="sf-icon-"], .step-default [class*=" sf-icon-"] {
      font-family: 'Default' !important;
      speak: none;
      font-style: normal;
      font-weight: normal;
      font-variant: normal;
      text-transform: none;
      line-height: inherit;
      -webkit-font-smoothing: antialiased;
      -moz-osx-font-smoothing: grayscale;
  }
  .step-default .sf-icon-cart:before { content: "\e710"; }
  .step-default .sf-icon-transport:before { content: "\e702"; }
  .step-default .sf-icon-payment:before { content: "\e706"; }
  .step-default .sf-icon-success:before { content: "\e715"; }

</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/defaultStepType" %}

Label type

In label type, the Stepper displays the steps with only the step labels by setting the stepType property as Label.

When both label and text are defined, the label takes priority in displaying the steps.

APP.VUE

```
<template>
  <div class="step-label">
    <ejs-stepper id="stepper" stepType="label">
        <e-steps>
          <e-step label='Cart' iconCss='sf-icon-cart'></e-step>
          <e-step label='Delivery' iconCss='sf-icon-transport'></e-step>
          <e-step label='Payment' iconCss='sf-icon-payment'></e-step>
          <e-step label='Ordered' iconCss='sf-icon-success'></e-step>
        </e-steps>
  </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
    components: {
      'ejs-stepper': StepperComponent,
      "e-steps": StepsDirective,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      "e-step": StepDirective
    }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";

  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgQEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
```

```
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxAQExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdABhAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  .step-label [class^="sf-icon-"], .step-label [class*=" sf-icon-"] {
      font-family: 'Default' !important;
      speak: none;
      font-style: normal;
      font-weight: normal;
      font-variant: normal;
      text-transform: none;
      line-height: inherit;
      -webkit-font-smoothing: antialiased;
      -moz-osx-font-smoothing: grayscale;
  }
  .step-label .sf-icon-cart:before { content: "\e710"; }
  .step-label .sf-icon-transport:before { content: "\e702"; }
  .step-label .sf-icon-payment:before { content: "\e706"; }
  .step-label .sf-icon-success:before { content: "\e715"; }

</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/labelStepType" %}

Label positions

You can display the label on the top, bottom, left, or right side of the steps using the labelPosition property.

The following label positions are supported in Stepper:

| Value | Description |

|-----|-----|

| Top | Positions the label at the top of each step. |

| Bottom | Positions the label at the bottom of each step. |

Copyright © 2001 -2024 Syncfusion Inc.

| Start | Positions the label to the left side of each step. |

| End | Positions the label to the right side of each step. |

APP.VUE

```
<template>
  <div class="step-label-position">
    <div class="e-btn-group">
      <input type="radio" id="start" name="position" value="start" v-
on:click="updateLabelPosition" />
      <label class="e-btn" for="start">Start</label>
      <input type="radio" id="end" name="position" value="end" v-
on:click="updateLabelPosition" checked />
      <label class="e-btn" for="end">End</label>
      <input type="radio" id="top" name="position" value="top" v-
on:click="updateLabelPosition" />
      <label class="e-btn" for="top">Top</label>
      <input type="radio" id="bottom" name="position" value="bottom" v-
on:click="updateLabelPosition" />
      <label class="e-btn" for="bottom">Bottom</label>
    </div>
    <div class="label-stepper-control">
      <ejs-stepper id="stepper" ref="stepper" labelPosition="end"
:activeStep="value">
        <e-steps>
          <e-step label='Cart' iconCss='sf-icon-cart'></e-step>
          <e-step label='Delivery Address' iconCss='sf-icon-transport'></e-
step>
          <e-step label='Payment' iconCss='sf-icon-payment'
optional=true></e-step>
          <e-step label='Confirmation' iconCss='sf-icon-success'></e-step>
        </e-steps>
      </ejs-stepper>
    </div>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  data() {
    return {
      value: 1
    };
  },
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  },
  methods: {
    updateLabelPosition: function (args) {
      this.$refs.stepper.ej2Instances.labelPosition =
args.currentTarget.value;
    }
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggfFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgQEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
```

```
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxAQExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  .sf-icon-cart:before { content: "\e710"; }
  .sf-icon-transport:before { content: "\e702"; }
  .sf-icon-payment:before { content: "\e706"; }
  .sf-icon-success:before { content: "\e715"; }
  .label-stepper-control {
    margin: 30px auto;
  }
  .step-label-position {
    text-align: center;
  }
  .step-label-position {
    margin-top: 30px;
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/labelPosition" %}

Indicator type

In indicator type, the Stepper displays steps with only the step indicators by setting the stepType property as Indicator.

APP.VUE

```
<template>
  <div class="step-indicator">
    <ejs-stepper id="stepper" stepType="indicator">
        <e-steps>
          <e-step></e-step>
          <e-step></e-step>
          <e-step></e-step>
          <e-step></e-step>
        </e-steps>
    </ejs-stepper>
    <ejs-stepper id="stepper1" stepType="indicator">
        <e-steps>
          <e-step text="1"></e-step>
          <e-step text="2"></e-step>
          <e-step text="3"></e-step>
          <e-step text="4"></e-step>
        </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
    components: {
      'ejs-stepper': StepperComponent,
      "e-steps": StepsDirective,
      "e-step": StepDirective
    }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  .step-indicator {
    margin-top: 30px;
    padding: 30px;
  }
  #stepper1 {
    margin-top: 50px;
  }

</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/indicatorType" %}

Copyright © 2001 -2024 Syncfusion Inc.

Linear flow in Vue Stepper component

The Stepper component enables users to progress sequentially through each step, ensuring navigation from one step to the next in a linear way by setting the linear property to true. The default value is false allowing navigation between any steps and vice versa.

The example demonstrates the functionality of both linear and non-linear flow in the Stepper.

APP.VUE

```
<template>
  <div class="step-type">
    <div class="e-btn-group">
      <input type="radio" id="linear" name="Linear"  value="true" v-
on:click="updateLinear" checked/>
      <label class="e-btn" for="linear">Linear</label>
      <input type="radio" id="nonLinear" name="Linear" value="false" v-
on:click="updateLinear" />
      <label class="e-btn" for="nonLinear">Non-Linear</label>
    </div>
    <div class="linear-stepper-control">
      <ejs-stepper id="stepper" ref="stepper" linear="true">
        <e-steps>
          <e-step label='Cart' iconCss='sf-icon-cart'></e-step>
          <e-step label='Delivery Address' iconCss='sf-icon-transport'></e-
step>
          <e-step label='Payment' iconCss='sf-icon-payment'
optional=true></e-step>
          <e-step label='Confirmation' iconCss='sf-icon-success'></e-step>
        </e-steps>
      </ejs-stepper>
    </div>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  },
  methods: {
      updateLinear: function (args) {
        this.$refs.stepper.ej2Instances.linear =
(/true/).test(args.currentTarget.value) ? true : false;
        this.$refs.stepper.ej2Instances.reset();
      }
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-
splitbuttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
```

```
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRyYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  .sf-icon-cart:before { content: "\e710"; }
  .sf-icon-transport:before { content: "\e702"; }
  .sf-icon-payment:before { content: "\e706"; }
  .sf-icon-success:before { content: "\e715"; }
  .linear-stepper-control {
    margin: 30px auto;
  }
  .step-type {
    text-align: center;
  }
  .step-type {
    margin-top: 30px;
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/linear" %}

Steps validation in Vue Stepper component

The Stepper component allows you to set the validation state for each step, displaying either a success or error icon. You can define the success state of a step by setting the `isValid` property to `true`. If set to `false`, the step will display an error state. By default, the `isValid` property is `null`.

Based on the `stepType`, the validation state icon will be displayed either as an indicator or as part of the step label/text.

APP.VUE

```
<template>
  <div class="step-validation">
```

```
    <ejs-stepper id="stepper">
      <e-steps>
        <e-step iconCss='sf-icon-cart' :isValid=true></e-step>
        <e-step iconCss='sf-icon-transport'></e-step>
        <e-step iconCss='sf-icon-payment' :isValid=false></e-step>
        <e-step iconCss='sf-icon-success'></e-step>
      </e-steps>
    </ejs-stepper>
    <ejs-stepper id="labelStep">
      <e-steps>
        <e-step label='Cart' :isValid=true></e-step>
        <e-step label='Address'></e-step>
        <e-step label='Payment' :isValid=false></e-step>
        <e-step label='Confirmation'></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
```

```
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
```

```
  }
  .sf-icon-cart:before { content: "\e710"; }
  .sf-icon-transport:before { content: "\e702"; }
  .sf-icon-payment:before { content: "\e706"; }
  .sf-icon-success:before { content: "\e715"; }
  .step-validation {
    margin-top: 30px;
    padding: 30px;
  }
  #labelStep {
    margin-top: 50px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/validation" %}

Template in Vue Stepper component

The Stepper component allows you to customize the default appearance and content of each step, creating a personalized experience for the user. You can use the template propety to set the template content for the steps.

The step model and current step index are passed as `step` and `currentStep` properties in the template context for customization.

APP.VUE

```
<template>
  <div class="stepper-template-wrapper">
    <ejs-stepper id="stepper" :activeStep=1 :template="tempContent">
      <e-steps>
        <e-step label='PowerPoint' iconCss='sf-icon-powerpoint'></e-step>
        <e-step label='Presentation' iconCss='sf-icon-projector'></e-step>
        <e-step label='Backup' iconCss='sf-icon-onedrive'></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  },
  data() {
    return {
      tempContent: '<div class="template-content"><span
class="${step.iconCss}"></span><br><span class="e-
label">${step.label}</span></div>'
    };
  }
};
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  .template-content {
        background: #fff;
        width: 65px;
  }
  /* Stepper progressbar customization */
  .e-stepper .e-stepper-progressbar {
      height: 3px;
      top: 25px;
  }
  .e-stepper .e-stepper-progressbar .e-progressbar-value {
      background-color: #27d96d;
  }
  /* Stepper status customization */
  #stepper .e-step-completed * {
      color: #19cd60;
  }
  #stepper .e-step-inprogress * {
      color: #3479f3;
  }
  #stepper .e-step-notstarted * {
      color: #bdbdbd;
  }
  @font-face {
  font-family: 'template_updated';
  src:
  url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tSfUAAAEoAAAAVmNtYXDnE+dkAAABlAAAADxnbHlm39z
zMQAAAdwAAAacaGVhZCaImHMAAADQAAAANmhoZWEIUQQGAAAArAAAACRobXR4FAAAAAAAAYAAAAA
UbG9jYQR8BVAAAAHQAAAADG1heHABFwEbAAABCAAAACBuYW1ldkXdggAACHgAAAKRcG9zdD2fuhI
AAAsMAAAAXwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAABQABAAAAAQAApDfGf18
PPPUACwQAAAAAAOG7KcEAAAAA4bspwQAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAFAQ8ACAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAAEAAgADAAQAAAAAAVQCAgMoA04ACAAAAAAD0wP0AB8APABcAHwA+gD+AQoBDgAAAQ8
HLwY9AT8GHwYlDwcvBj0BPwI7AR8FNw8HLwY9AT8GHwYlDwYrAS8GPwczHwUHFR8HMzcXDwIfBz8
HJzcfAz8CFw8BHwc/By8HDwMnPwI1LwcPBxUfAQcvBCMPAic/Ay8HDwYlESERAyEHFzczFzcnIRE
hJyE1IQMiAQECAwQEBQUFBAUDAwICAgIDAwUEBQUFBAQDAgH+mwEBAgMEBAUFBQQFAwMCAgILBgU
FBQQEAwIBnAEBAgMEBAUFBQQEBAMCAgICAwQEBAUFBQQEAwIB/rkBAQMDAwUEBQUFBAQDAgEBAQE
CAwQEBQUFBAUDAwMBUAEDBgYJCQsLCAxoAgQBAQQFBwgJCwsLCwkJBgYDAQFECAgICAkIBIEBAQE
DBQcICgoLDAoJCQcFAwEBAwUHCQkKDAoJCQl4BQMCAQMFBwgKCgwLCgoIBwUDAQICPwEJCQoMBwc
GCGQFAwIBAQQFBwgJCwsLCwkJBgUEAsb89j8BZ8sr+gX7K8sBZ/x4DwOm/FoB7QUFBAQDAgEBAQE
CAwQEBQUFBAUDAwMBAQEBAwMDBQQzBQUEBAMCAQEBAQIDBAQFBQUGCwICAgMEBARgBQUEAwMDAQE
BAQMDAwQFBQUFBAMDAwEBAQEDAwMEBUsFBAQEAwICAgIDBAQEBQUFBAQDAgEBAgIDBAQFBQUGCgo
IBwUDAQNnBAsLCwsJCQYFBAEBAwYGCQkLCwkuBAMCAQECAVkFCwsLCQgHBQQBAQQFBwgJCwsLCwk
JBgYDAQECBQZTCAcJCAsLCQgHBQQBAQQFBwgJCwsJCAUqAgcFAwEBAgRkCQgICAwKCggHBQMBAQM
FBwgKCmP97gIS/bDALe3tLcACji8+AAAGAAAAAAP0A+QAEwA5AEsAdwCFAIkAAAEzPwc1LwcjFxU
PDisBFSMRNx8OBSM1Mx8NJz8CFTMVDw4vAxUzFSMVMxUjFSE/AhEvAiE1IREzPwMRLwMhJREFEQE
QIgcHBgUFBAECAQIEBQUGBAYmiAECAgQEBQYGCAcJCAkKCQokNlsMCwoKCQgHBwYFBAMDAgEBgn0
QDAwLCwoJCQgHBgUEAwK7BA4NnAIEBAYHCAkJCgsMDAwNDQcWDgza2traAU0FAwICAwX+swF3FQQ
DAgEBAgMC/nL9rgIyAgABAgQFBgYDByQHBgYFBQQBAiEfCQoJCQkIBwcGBgUEAwICcAFIBQEBAgM
```

```
DBQUGBgcICAkKCgV9AgMEBgYHCAkJCwsLDA1QAgUBhA0NDAsKCgoIBwcGBQQCAgEBAgMELSA+Hz8
CBAQB/wUDAiD+SgECAwIBwwQDAgFb/PdfA8gAAAACAAAAAAPzAykAcQEIAAABDyAVHw4lPw01Lwo
1LxErAQ8CKwEvCQcnDwgvBSsCDxQVDw4VHw8zLwQ1Px8fCj8ELxMPAgICCwwLCgoKCQkLCggIBgU
EAwECEg8QDg0LCgkIBgUEAgEBAQIEBQYICQoLDQ4PDg4PAh0ODgYHBgcHBQUFAwMDBAYDBgcICQo
MDBMCAgQCBAQFBgYGBwcHCAgJCRIUEwoSEhQCARYICw0NDQ4PDw8XFwoVExIQDw4NCQwLCwsLCws
LCwoLCwoLCgsJCggHBwYGBQQDAwIBARAPDgwLCgkHBwYEBAMBAQECBAUGBwgJCgoKCgoLCwtgCwY
FAwICAwQGBwkKCw0NDg8QEgIBBAMHBwgKCwwNDxITExMUFBUUDw8ODQwMCwsJCAgJERsFBAQFBgY
HCAgJCgoLDAwNFBUUFhELCwKEAgMEBQUHBwgNDQ0PDw8QEgIBAgMFBgYICAkKCwwOCgsKCgsKFA8
NDAsKCQcGBAMBAQEBBAMDBAYGBwcICAkIGBIdCA0LCggHBgUGAhAREQkICAkICAcHBgYFBAQDBQM
DBQgbBwoJCAYFAwIBAZ0DBwkLDQ4QEg8GBAQDAgICAgQEBQYGCAcICQkJCgkLCgsLDAsZAwQGBgc
ICAkKCwsLDAwMDA0NDQsMCgoKCQcGBQQDAwEUEBAQEBEQEQ8PDQwLCgkJBwYFBQECDgwREA8ODgw
LCwoIBgUCAQEDBAUGBwgJCwsMBAMDAwEZDQ0NDAsLCwoJCQgICAYGCAYEAgEBAgADAAAAAAAPfA/Q
ACwAPABMAAAEHFzcVMzUXNyc1IyUhESEnITUhAdq+NYlLijW/S/6JAzX8y0MDwPxAAQC/NYp+foo
1v1ErAc5DZwAAAAASAN4AAQAAAAAAAAABAAAAAQAAAAAAQAQAAEAAQAAAAAAAgAHABEAAQAAAAA
AAwAQABgAAQAAAAAABAAQACgAAQAAAAAABQALADgAAQAAAAAABgAQAEMAAQAAAAAACgAsAFMAAQA
AAAAACwASAH8AAwABBAkAAAACAJEAAwABBAkAAQAgAJMAAwABBAkAAgAOALMAAwABBAkAAwAgAME
AAwABBAkABAAgAOEAAwABBAkABQAWAQEAAwABBAkABgAgARcAAwABBAkACgBYATcAAwABBAkACwA
kAY8gdGVtcGxhdGVfdXBkYXRlZFJlZ3VsYXJ0ZW1wbGF0ZV91cGRhdGVkdGVtcGxhdGVfdXBkYXR
lZFZlcnNpb24gMS4wdGVtcGxhdGVfdXBkYXRlZEZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXN
pb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAdABlAG0AcABsAGEAdABlAF8AdQB
wAGQAYQB0AGUAZABSAGUAZwB1AGwAYQByAHQAZQBtAHAAbABhAHQAZQBfAHUAcABkAGEAdABlAGQ
AdABlAG0AcABsAGEAdABlAF8AdQBwAGQAYQB0AGUAZABWAGUAcgBzAGkAbwBuACAAMQAuADAAdAB
lAG0AcABsAGEAdABlAF8AdQBwAGQAYQB0AGUAZABGAG8AbgB0ACAAZwBlAG4AZQByAGEAdABlAGQ
AIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZAB
pAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAAAAAAKAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgANcHJvamVjdG9yLW9sZApwb3dlcnBvaW50CG9
uZWRyaXZlDXByb2plY3Rvci1uZXcAAAA=) format('truetype');
  font-weight: normal;
  font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
  font-family: 'template_updated' !important;
  speak: none;
  font-size: 40px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  }
  .sf-icon-projector:before { content: "\e700"; }
  .sf-icon-powerpoint:before { content: "\e701"; }
  .sf-icon-onedrive:before { content: "\e702"; }
  .stepper-template-wrapper {
      width: 75%;
      margin: 0px auto;
      min-width: 85px;
      padding: 25px 0;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/template" %}

Copyright © 2001 -2024 Syncfusion Inc.

Tooltip in Vue Stepper component

The Stepper component supports tooltip to show additional information in the steps by setting the showTooltip property to true.

The tooltip appears when the user hovers over the step, providing the information such as the label or text. By default, the showTooltip property is false.

APP.VUE

```
<template>
  <div class="step-tooltip">
    <ejs-stepper id="stepper" showTooltip="true">
      <e-steps>
        <e-step label='Cart' iconCss='sf-icon-cart'></e-step>
        <e-step label='Shipped' iconCss='sf-icon-transport'></e-step>
        <e-step label='Payment' iconCss='sf-icon-transport'></e-step>
        <e-step label='Delivered' iconCss='sf-icon-success'></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw



```

```
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCgoJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
```

```
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  .sf-icon-cart:before { content: "\e710"; }
  .sf-icon-transport:before { content: "\e702"; }
  .sf-icon-payment:before { content: "\e706"; }
  .sf-icon-success:before { content: "\e715"; }
  .step-tooltip {
    margin-top: 30px;
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/tooltip" %}

Tooltip template

You can use the tooltipTemplate property to specify a custom template for the tooltips, providing detailed information about the steps.

APP.VUE

```
<template>
  <div class="step-tooltip-template">
    <ejs-stepper id="stepper" showTooltip="true"
:tooltipTemplate="tooltipContent">
      <e-steps>
        <e-step text='Review your cart items' label='Cart' iconCss='sf-icon-
cart'></e-step>
        <e-step text='Enter your delivery address' label='Delivery Address'
iconCss='sf-icon-transport'></e-step>
        <e-step text='Complete your purchase securely' label='Payment'
iconCss='sf-icon-payment'></e-step>
        <e-step text='Verify your order details' label='Confirmation'
iconCss='sf-icon-success'></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  },
  data() {
    return {
      tooltipContent: '<span class="template-content"><span class="stepper-
icon ${value.iconCss}"></span><span class="step-
label">${value.text}</span></span>'
    };
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  .step-tooltip-template {
    margin-top: 30px;
    padding: 30px;
  }
  .template-content {
      display: flex;
      align-items: center;
  }
  .e-stepper-tooltip {
      border-radius: 5px;
  }
  .e-stepper-tooltip .e-tip-content {
      padding: 7px;
  }
  .step-label {
      margin-left: 7px;
      font-weight: bold;
  }
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
```

```
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRyby BTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdABhAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  .sf-icon-cart:before { content: "\e710"; }
  .sf-icon-transport:before { content: "\e702"; }
  .sf-icon-payment:before { content: "\e706"; }
  .sf-icon-success:before { content: "\e715"; }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/tooltipTemplate" %}

Copyright © 2001 -2024 Syncfusion Inc.

Animation in Vue Stepper component

The Stepper progress state can be animated, smoothly transitioning from one step to another. You can customize the animation's duration and delay by using the animation property.

You can disable the animation by setting the enable property to false. By default, the value is true.

| Fields | Type | Description |

|------|------|-------------|

| duration | number | Specifies the duration of the animated transition for each step. The default value is 2000 milliseconds. |

| delay | number | Specifies the delay to initiate the animated transition for each step in milliseconds. The default value is 0. |

The example demonstrates the animation duration and delay settings for the Stepper.

APP.VUE

```
<template>
  <div class="animation-container">
    <ejs-stepper id="stepper" :animation="animation">
      <e-steps>
        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  },
  data() {
    return {
      animation: { enable:true, delay: '500', duration: '2000' }
    };
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  .animation-container {
    margin-top: 30px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/animation" %}

Globalization in Vue Stepper component

Localization

The Localization library allows you to localize the default text content of the Stepper. You can change the static text content used for the optional property to other cultures (Arabic, Deutsch, French, etc.) by defining the locale value and its translation object.

Locale key | en-US (default) |

|-----|-----|

| optional | Optional |

In this example, the French culture is set to Stepper and the default text is updated with the content defined by the locale key.

APP.VUE

```
<template>
  <div class="step-locale">
    <ejs-stepper id="stepper" :locale='locale'>
        <e-steps>
          <e-step label='Cart' iconCss='sf-icon-cart'></e-step>
          <e-step label='Delivery Address' iconCss='sf-icon-transport'></e-
step>
          <e-step label='Payment' iconCss='sf-icon-payment'
optional=true></e-step>
          <e-step label='Confirmation' iconCss='sf-icon-success'></e-step>
        </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
import { L10n } from '@syncfusion/ej2-base';
L10n.load({
  'fr-BE': {
      'stepper': {
      'optional': "facultatif"
    }
  }
});
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  },
  data() {
    return {
```

```
      locale: 'fr-BE'
    }
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgQEBQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
```

```
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  .sf-icon-cart:before { content: "\e710"; }
  .sf-icon-transport:before { content: "\e702"; }
  .sf-icon-payment:before { content: "\e706"; }
  .sf-icon-success:before { content: "\e715"; }
  .step-locale {
    margin-top: 30px;
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/localization" %}

RTL

RTL provides an option to switch the text direction and layout of the Stepper component from right to left by setting the enableRtl property to true.

APP.VUE

```
<template>
  <div class="rtl-step">
    <ejs-stepper id="stepper" enableRtl="true">
        <e-steps>
```

```
          <e-step text='Cart' iconCss='sf-icon-cart'></e-step>
          <e-step text='Delivery Address' iconCss='sf-icon-transport'></e-
step>
          <e-step text='Payment' iconCss='sf-icon-payment'
optional=true></e-step>
          <e-step text='Confirmation' iconCss='sf-icon-success'></e-step>
        </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  @font-face {
    font-family: 'Default';
    src:
    url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgcHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgcHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/l0ICAcGBQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
```

```
IERESFBQUFQQDAgECAgMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgoHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COwEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAwUGBwgJCgICAQENAQEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQQnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDAwQECA8XPRcKCgUPFz0
REAkIBAMDAwMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAAABAAcAAQABAAAAAAAACAAcACAABAAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAAFAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRyYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIARABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
  [class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  .sf-icon-cart:before { content: "\e710"; }
  .sf-icon-transport:before { content: "\e702"; }
  .sf-icon-payment:before { content: "\e706"; }
  .sf-icon-success:before { content: "\e715"; }
  .rtl-step {
    margin-top: 30px;
    padding: 30px;
```

```
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/rtl" %}

Accessibility in Vue Stepper component

Accessibility is achieved in the Stepper component through WAI-ARIA standard and keyboard navigations. The Stepper component can be effectively accessed through assistive technologies such as screen readers.

Keyboard interaction

The Stepper component is interactive with the below keyboard shortcuts.

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Up Arrow | Focuses the previous step in a vertical Stepper. |

| Down Arrow | Focuses the next step in a vertical Stepper. |

| Left Arrow | Focuses the previous step in a horizontal Stepper. |

| Right Arrow | Focuses the next step in a horizontal Stepper. |

| Home | Focuses on the first step of the Stepper. |

| End | Focuses on the last step of the Stepper. |

| Enter / Space | Activates the currently focused step. |

ARIA attribute

The following ARIA attributes are used in Stepper component based on its state.

| Properties | Functionality |

| ---------- | ---------- |

| aria-label | Attribute provide a descriptive text that labels the interactive element for accessibility. |

| aria-current | Attribute denotes the currently active step in the Stepper. |

| aria-disabled| Indicates when the step is disabled and cannot be interacted. |

Events in Vue Stepper component

This section describes the Stepper events that will be triggered when an appropriate actions are performed. The following events are available in the Stepper component.

created

The Stepper component triggers the created event when the control rendering is completed.

APP.VUE

```
<template>
  <div class="created-event">
    <ejs-stepper id="stepper" :created="stepCreated">
      <e-steps>
        <e-step></e-step>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  },
  methods: {
    stepCreated: function(args) {
      //Your required action here
    }
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  .created-event {
    margin-top: 30px;
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/step-created" %}

stepChanged

The Stepper component triggers the stepChanged event after the active step is changed.

APP.VUE

```
<template>
  <div class="stepChanged-event">
    <ejs-stepper id="stepper" :stepChanged="stepChanged">
      <e-steps>
        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
```

```
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  },
  methods: {
    stepChanged: function(args) {
      alert("Step Changed from " + args.previousStep + " to " +
args.activeStep)
    }
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  .stepChanged-event {
    margin-top: 30px;
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/step-changed" %}

stepChanging

The Stepper component triggers the stepChanging event before the active step change.

APP.VUE

```
<template>
  <div class="stepChanging-event">
    <ejs-stepper id="stepper" :stepChanging="stepChanging">
      <e-steps>
        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
  methods: {
    stepChanging: function(args) {
      alert("Step changing from " + args.previousStep + " to " +
args.activeStep)
    }
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
  .stepChanging-event {
    margin-top: 30px;
    padding: 30px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stepper/step-changing" %}

stepClick

The Stepper component triggers the stepClick event when the step is clicked.

APP.VUE

```
<template>
  <ejs-stepper id="stepper" :stepClick="stepClick">
    <e-steps>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
    </e-steps>
  </ejs-stepper>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  },
  methods: {
    stepClick: function(args) {
      //Your required action here
    }
  }
};
</script>
<style>
```

```
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
</style>
```

beforeStepRender

The Stepper component triggers the beforeStepRender event before rendering each step.

APP.VUE

```
<template>
  <ejs-stepper id="stepper" :beforeStepRender="beforeStepRender">
    <e-steps>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
    </e-steps>
  </ejs-stepper>
</template>
<script>
import { StepperComponent, StepsDirective, StepDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-stepper': StepperComponent,
    "e-steps": StepsDirective,
    "e-step": StepDirective
  },
  methods: {
    beforeStepRender: function(args) {
      //Your required action here
    }
  }
};
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-base/styles/material.css";
</style>
```

Stock Chart

Getting Started with the Vue Stock chart Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Stock chart component

Copyright © 2001 -2024 Syncfusion Inc.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The list of minimum dependencies required to use stock chart are follows:

```
|-- @syncfusion/ej2-charts
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-compression
|-- @syncfusion/ej2-svg-base
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-calendars
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-buttons
|-- @syncfusion/ej2-splitbuttons
```

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

```bash
npm install -g @vue/cli
vue create quickstart
cd quickstart
npm run serve
```

or

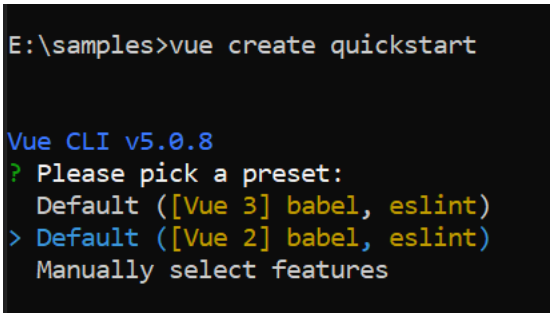
```bash
yarn global add @vue/cli
vue create quickstart
cd quickstart
```

Copyright © 2001 -2024 Syncfusion Inc.

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Stock chart component as an example. Install the @syncfusion/ej2-vue-charts package by running the following command:

`bash

npm install @syncfusion/ej2-vue-charts --save

`

or

`bash

yarn add @syncfusion/ej2-vue-charts

`

The **--save** will instruct NPM to include the chart package inside of the dependencies section of the package.json.

Add Syncfusion Vue component

Follow the below steps to add the Vue Stock chart component:

1\. First, import and register the Stock chart component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { StockChartPlugin } from '@syncfusion/ej2-vue-charts';
export default {
components: {
'ejs-stockchart': StockChartComponent
}
}
</script>
```

2\. In the template section, define the Stock chart component.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-stockchart></ejs-stockchart>
</div>
</template>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

Module Injection

To create stock chart with additional features, inject the required modules. The following modules are used to extend stock chart's basic functionality.

•CandleSeries - Inject this module to use candle series.

•DateTime - Inject this module to use date time axis.

•RangeTooltip - Inject this module to show the tooltip.

These modules should be injected to the provide section as follows,

`javascript

<script>

import { StockChartComponent, CandleSeries, DateTime, RangeTooltip } from "@syncfusion/ej2-vue-charts";

export default {

components: {

'ejs-stockchart': StockChartComponent

},

provide: {

stockChart: [CandleSeries, DateTime, RangeTooltip]

}

Copyright © 2001 -2024 Syncfusion Inc.

```
};
</script>
`
```

Populate Stock Chart with Data

This section explains how to plot below JSON data to the Stock Chart. Please find the below imported datasource.

```
`javascript
export default {
data() {
return {
data: [{
x: new Date( '2012-04-02' ),
open : 85.9757,
high : 90.6657,
low : 85.7685,
close : 90.5257,
volume : 660187068
},
{
x: new Date( '2012-04-09' ),
open : 89.4471,
high : 92,
low : 86.2157,
close : 86.4614,
volume : 912634864
},
{
x: new Date( '2012-04-16' ),
open : 87.1514,
high : 88.6071,
low : 81.4885,
close : 81.8543,
volume : 1221746066
},
```

{

x: new Date('2012-04-23'),

open : 81.5157,

high : 88.2857,

low : 79.2857,

close : 86.1428,

volume : 965935749

},

{

x: new Date('2012-04-30'),

open : 85.4,

high : 85.4857,

low : 80.7385,

close : 80.75,

volume : 615249365

}];

};

}

};

`

Add a series object to the Stock chart by using series property and then set the JSON data to dataSource property.

Since the JSON contains category data, set the valueType for horizontal axis to Category. By default, the axis valueType is Numeric.

~/SRC/APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
```

```
  </div>
</template>
<script>
import { chartData } from "./datasource.js";
import {
  StockChartComponent, StockChartSeriesCollectionDirective,
StockChartSeriesDirective, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
export default {
  components: {
    'ejs-stockchart': StockChartComponent,
    'e-stockchart-series-collection': StockChartSeriesCollectionDirective,
    'e-stockchart-series': StockChartSeriesDirective
  },
  data() {
    return {
        seriesData: chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/stockchart/getting-started-cs1" %}

Add Stock Chart Title

You can add a title using title property to the Stock Chart to provide quick information to the user about the data plotted in the Chart.

~/SRC/APP.VUE

```
<template>
    <div id="app">
```

```
        <ejs-stockchart :valueType='valueType' :labelFormat='labelFormat'
:title='title'>
            <e-stockchart-series-collection>
                <e-stockchart-series :dataSource='seriesData' type='Candle'
xName='date' low='low' high='high' open='open' close='close' width=2>
                </e-stockchart-series>
            </e-stockchart-series-collection>
        </ejs-stockchart>
    </div>
</template>
<script>
import { StockChartComponent, StockChartSeriesCollectionDirective,
StockChartSeriesDirective, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export} from "@syncfusion/ej2-vue-
charts";
import { chartData } from "./datasource.js";
export default {
  components: {
    'ejs-stockchart': StockChartComponent,
    'e-stockchart-series-collection': StockChartSeriesCollectionDirective,
    'e-stockchart-series': StockChartSeriesDirective
  },
  data() {
    return {
     valueType: 'DateTime',
     seriesData: chartData,
     labelFormat: 'MMM-yy',
     title: 'Sales Analysis'
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator, Export
    ]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/stockchart/getting-started-cs2" %}

Add Crosshair

Crosshair has a vertical and horizontal line to view the value of the axis at mouse or touch position.

Crosshair lines can be enabled by using enable property in the crosshair. Likewise tooltip label for an axis can be enabled by using enable property of crosshairTooltip in the corresponding axis.

~/SRC/APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div id="app">
        <ejs-stockchart :title='title'  :crosshair="crosshair">
            <e-stockchart-series-collection>
                <e-stockchart-series :dataSource='seriesData' type='Candle'>
                </e-stockchart-series>
            </e-stockchart-series-collection>
        </ejs-stockchart>
    </div>
</template>
<script>
import { chartData } from "./datasource.js";
import {
  StockChartComponent, StockChartSeriesCollectionDirective,
StockChartSeriesDirective, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,Crosshair,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
export default {
  components: {
    'ejs-stockchart': StockChartComponent,
    'e-stockchart-series-collection': StockChartSeriesCollectionDirective,
    'e-stockchart-series': StockChartSeriesDirective
  },
  data() {
    return {
     title: 'Sales Analysis',
     seriesData: chartData,
       crosshair: {
        enable: true,
      }
    };
  },
   provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,Crosshair,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator, Export
    ]
  }
};
</script>
```

{% previewsample "page.domainurl/code-snippet/stockchart/getting-started-cs3" %}

Add Trackball

Trackball is used to track a data point closest to the mouse or touch position. Trackball marker indicates the

closest point and trackball tooltip displays the information about the point. To use trackball feature, we need to inject Crosshair and Tooltip into the provide.

Trackball can be enabled by setting the enable property of the crosshair to true and shared property in tooltip to true in chart.

~/SRC/APP.VUE

```
<template>
    <div id="app">
         <ejs-stockchart id= "container" :title='title'
:primaryXAxis='primaryXAxis' :crosshair='crosshair' :tooltip='tooltip'>
            <e-stockchart-series-collection>
                <e-stockchart-series :dataSource='seriesData' type='Line'
xName='x' yName='y' name='John' > </e-stockchart-series>
                <e-stockchart-series :dataSource='seriesData' type='Line'
xName='x' yName='y1' name='Andrew' > </e-stockchart-series>
                <e-stockchart-series :dataSource='seriesData' type='Line'
xName='x' yName='y2' name='Thomas' > </e-stockchart-series>
                <e-stockchart-series :dataSource='seriesData' type='Line'
xName='x' yName='y3' name='Mark' > </e-stockchart-series>
                <e-stockchart-series :dataSource='seriesData' type='Line'
xName='x' yName='y4' name='William' > </e-stockchart-series>
            </e-stockchart-series-collection>
        </ejs-stockchart>
    </div>
</template>
<script>
import {
  StockChartComponent, StockChartSeriesCollectionDirective,
StockChartSeriesDirective, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,Crosshair,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, Tooltip,AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
export default {
  components: {
    'ejs-stockchart': StockChartComponent,
    'e-stockchart-series-collection': StockChartSeriesCollectionDirective,
    'e-stockchart-series': StockChartSeriesDirective
  },
  data() {
    return {
        seriesData: [
            { x: new Date(2000, 2, 11), y: 15, y1: 39, y2: 60, y3: 75, y4:
85 },
            { x: new Date(2000, 9, 14), y: 20, y1: 30, y2: 55, y3: 75, y4:
83 },
            { x: new Date(2001, 2, 11), y: 25, y1: 28, y2: 48, y3: 68, y4:
85 },
            { x: new Date(2001, 9, 16), y: 21, y1: 35, y2: 57, y3: 75, y4:
87 },
            { x: new Date(2002, 2, 7), y: 13, y1: 39, y2: 62, y3: 71, y4: 82
},
```

```
            { x: new Date(2002, 9, 7), y: 18, y1: 41, y2: 64, y3: 69, y4: 74
},
            { x: new Date(2003, 2, 11), y: 24, y1: 45, y2: 57, y3: 81, y4:
73 },
            { x: new Date(2003, 9, 14), y: 23, y1: 48, y2: 53, y3: 84, y4:
75 },
            { x: new Date(2004, 2, 6), y: 19, y1: 54, y2: 63, y3: 85, y4: 73
},
            { x: new Date(2004, 9, 6), y: 31, y1: 55, y2: 50, y3: 87, y4: 60
},
            { x: new Date(2005, 2, 11), y: 39, y1: 57, y2: 66, y3: 75, y4:
48 },
            { x: new Date(2005, 9, 11), y: 50, y1: 60, y2: 65, y3: 70, y4:
55 },
            { x: new Date(2006, 2, 11), y: 24, y1: 60, y2: 79, y3: 85, y4:
40 }
              ],
      primaryXAxis: {
            title: 'Years',
            minimum: new Date(2000, 1, 1), maximum: new Date(2006, 2, 11),
            intervalType: 'Months',
            valueType: 'DateTime',
        },
        title: "Average Sales per Person",
        crosshair: {  enable: true, lineType: 'Vertical' },
        tooltip: { enable: true, shared: true, format: '${series.name} :
${point.x} : ${point.y}' }
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,Tooltip,Crosshair,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator, Export
    ]
  }
};
</script>
<style>
 #container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/getting-started-cs4" %}

You can refer to our Vue Stock Chart feature tour page for its groundbreaking feature representations. You can also explore our Vue Stock Chart example that shows you how to present and manipulate data.

Getting Started with the Vue Stock Chart Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Stock Chart component using the Composition API / Options API.

Copyright © 2001 -2024 Syncfusion Inc.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

Copyright © 2001 -2024 Syncfusion Inc.

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Stock Chart component as an example. To use the Vue Stock Chart component in the project, the @syncfusion/ej2-vue-charts package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-charts --save

`

or

`bash

yarn add @syncfusion/ej2-vue-charts

`

Copyright © 2001 -2024 Syncfusion Inc.

Add Syncfusion Vue component

Follow the below steps to add the Vue Stock Chart component using Composition API or Options API:

1.First, import and register the Stock Chart component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { StockChartComponent as EjsStockchart,
StockChartSeriesCollectionDirective as EStockchartSeriesCollection,
StockChartSeriesDirective as EStockchartSeries, StockChart, DateTime,
RangeTooltip, LineSeries, SplineSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator, RsiIndicator,
BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator, AtrIndicator,
AccumulationDistributionIndicator, MacdIndicator, StochasticIndicator,
Export } from "@syncfusion/ej2-vue-charts";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { StockChartComponent, StockChartSeriesCollectionDirective,
StockChartSeriesDirective, StockChart, DateTime, RangeTooltip, LineSeries,
SplineSeries, CandleSeries, HiloOpenCloseSeries, HiloSeries,
RangeAreaSeries, Trendlines, EmaIndicator, RsiIndicator,
BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator, AtrIndicator,
AccumulationDistributionIndicator, MacdIndicator, StochasticIndicator,
Export } from "@syncfusion/ej2-vue-charts";
//Component registration
export default {
name: "App",
components: {
"ejs-stockchart": StockChartComponent,
"e-stockchart-series-collection": StockChartSeriesCollectionDirective,
"e-stockchart-series": StockChartSeriesDirective
}
}
</script>
```

2.In the template section, define the Stock Chart component with the dataSource property.

~/SRC/APP.VUE

```
<template>
<ejs-stockchart :primaryXAxis="primaryXAxis" :primaryYAxis="primaryYAxis"
:title="title">
<e-stockchart-series-collection>
<e-stockchart-series :dataSource="seriesData" type="Candle" volume="volume"
xName="date" low="low" high="high" open="open" close="close" yName="open">
</e-stockchart-series>
</e-stockchart-series-collection>
</ejs-stockchart>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

3.Declare the values for the dataSource property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const seriesData = [
{
date: new Date('2012-04-02'),
open: 85.9757,
high: 90.6657,
low: 85.7685,
close: 90.5257,
volume: 660187068
},
{
date: new Date('2012-04-09'),
open: 89.4471,
high: 92,
low: 86.2157,
close: 86.4614,
volume: 912634864
},
....
];
const primaryXAxis = { valueType: "DateTime" };
const primaryYAxis = { majorTickLines: { color: "transparent", width: 0 } };
const title = "AAPL Stock Price";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
seriesData: [
{
date: new Date('2012-04-02'),
open: 85.9757,
high: 90.6657,
low: 85.7685,
close: 90.5257,
volume: 660187068
},
{
date: new Date('2012-04-09'),
open: 89.4471,
high: 92,
low: 86.2157,
close: 86.4614,
volume: 912634864
},
....
],
primaryXAxis: {
valueType: "DateTime"
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
primaryYAxis: {
majorTickLines: { color: "transparent", width: 0 },
},
title: "AAPL Stock Price"
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-stockchart :primaryXAxis="primaryXAxis" :primaryYAxis="primaryYAxis"
:title="title">
<e-stockchart-series-collection>
<e-stockchart-series :dataSource="seriesData" type="Candle" volume="volume"
xName="date" low="low" high="high" open="open" close="close" yName="open">
</e-stockchart-series>
</e-stockchart-series-collection>
</ejs-stockchart>
</template>
<script setup>
import { provide } from 'vue';
import { StockChartComponent as EjsStockchart,
StockChartSeriesCollectionDirective as EStockchartSeriesCollection,
StockChartSeriesDirective as EStockchartSeries, StockChart, DateTime,
RangeTooltip, LineSeries, SplineSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator, RsiIndicator,
BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator, AtrIndicator,
AccumulationDistributionIndicator, MacdIndicator, StochasticIndicator,
Export } from "@syncfusion/ej2-vue-charts";
const seriesData = [
{
date: new Date('2012-04-02'),
open: 85.9757,
high: 90.6657,
low: 85.7685,
close: 90.5257,
volume: 660187068
},
{
date: new Date('2012-04-09'),
open: 89.4471,
high: 92,
low: 86.2157,
close: 86.4614,
volume: 912634864
},
....
];
const primaryXAxis = { valueType: "DateTime" };
const primaryYAxis = { majorTickLines: { color: "transparent", width: 0 } };
const title = "AAPL Stock Price";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
const stockChart = [ DateTime, RangeTooltip, LineSeries, SplineSeries,
CandleSeries, HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator, BollingerBands, TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator,Export ];
provide('stockChart', stockChart);
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-stockchart :primaryXAxis="primaryXAxis" :primaryYAxis="primaryYAxis"
:title="title">
<e-stockchart-series-collection>
<e-stockchart-series :dataSource="seriesData" type="Candle" volume="volume"
xName="date" low="low" high="high" open="open" close="close" yName="open">
</e-stockchart-series>
</e-stockchart-series-collection>
</ejs-stockchart>
</template>
<script>
import { StockChartComponent, StockChartSeriesCollectionDirective,
StockChartSeriesDirective, StockChart, DateTime, RangeTooltip, LineSeries,
SplineSeries, CandleSeries, HiloOpenCloseSeries, HiloSeries,
RangeAreaSeries, Trendlines, EmaIndicator, RsiIndicator,
BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator, AtrIndicator,
AccumulationDistributionIndicator, MacdIndicator, StochasticIndicator,
Export } from "@syncfusion/ej2-vue-charts";
//Component registration
export default {
name: "App",
components: {
"ejs-stockchart": StockChartComponent,
"e-stockchart-series-collection": StockChartSeriesCollectionDirective,
"e-stockchart-series": StockChartSeriesDirective
},
data() {
return {
seriesData: [
{
date: new Date('2012-04-02'),
open: 85.9757,
high: 90.6657,
low: 85.7685,
close: 90.5257,
volume: 660187068
},
{
date: new Date('2012-04-09'),
open: 89.4471,
high: 92,
low: 86.2157,
close: 86.4614,
volume: 912634864
},
....
```

Copyright © 2001 -2024 Syncfusion Inc.

```
],
primaryXAxis: {
valueType: "DateTime"
},
primaryYAxis: {
majorTickLines: { color: "transparent", width: 0 },
},
title: "AAPL Stock Price"
};
},
provide: {
stockChart: [ DateTime, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator, BollingerBands, TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export ]
},
};
</script>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

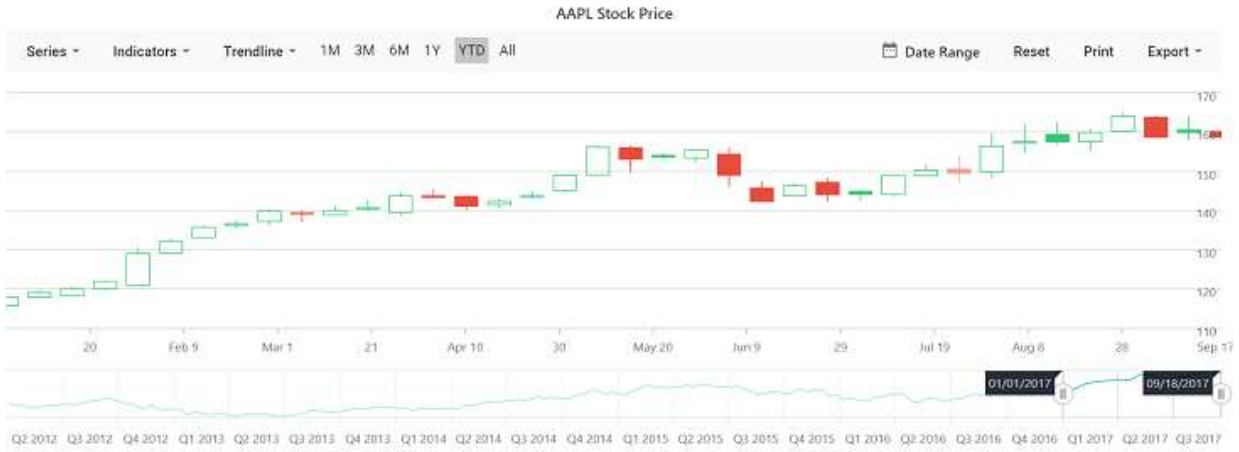
or

`bash

yarn run dev

`

The output will appear as follows:



Sample: vue-3-stock-chart-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

Copyright © 2001 -2024 Syncfusion Inc.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

<!-- markdownlint-disable MD036 -->

Working with data in Vue Stock chart component

Stock Chart can visualise data bound from local or remote data.

Local Data

You can bind a simple JSON data to the chart using dataSource property in series.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price',
```

```
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/working-with-data-cs1" %}

See Also

- Series Types

Chart dimensions in Vue Stock chart component

Size for Container

Stock Chart can render to its container size. You can set the size via inline or CSS as demonstrated below.

APP.VUE

```
<template>
  <div class="control-section" width='650px' height='350px'>
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/chart-dimensions-cs1" %}

Size for Stock Chart

You can also set size for stock chart directly through width and

height properties.

<!-- markdownlint-disable MD036 -->

- In Pixel

<!-- markdownlint-disable MD036 -->

You can set the size of stock chart in pixel as demonstrated below.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        width='650px'
        height='350px'
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/chart-dimensions-cs2" %}

In Percentage

By setting value in percentage, stock chart gets its dimension with respect to its container. For example, when the height is ‘50%’, stock chart renders to half of the container height.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
         width='80%' height='90%'
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export, Tooltip, Crosshair
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
```

```
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, Tooltip, Crosshair, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,     HiloOpenCloseSeries, HiloSeries, RangeAreaSeries,
Trendlines, EmaIndicator, RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/chart-dimensions-cs3" %}

Note: When you do not specify the size, it takes 450px as the height and window size as its width.

Axis types in Vue Stock chart component

DateTime axis

DateTime axis uses date time scale and displays the date time values as axis labels in the specified format. To use DateTime axis, set the valueType of axis to DateTime.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            lineStyle: { color: "transparent" },
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price'
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/axis-types-cs1" %}

Note: To use this axis, we need to inject DateTimeService into the provide and set the valueType of axis to DateTime.

DateTimeCategory axis

DateTimeCategory axis in the stock chart is used to display only business days. To use DateTimeCategory axis, set the valueType of axis to DateTimeCategory.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
```

```
        :crosshair="crosshair"
        :tooltip="tooltip">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Line"
xName='x' yName='y'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import {
  StockChartPlugin, DateTimeCategory, CandleSeries, RangeTooltip,
LineSeries, SplineSeries, Tooltip, Crosshair,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
let datetimeCategoryData = [
  { x: new Date(2021, 1, 11) }, { x: new Date(2021, 1, 12) }, { x: new
Date(2021, 1, 13) }, { x: new Date(2021, 1, 14) }, { x: new Date(2021, 1,
15) },
  { x: new Date(2021, 1, 19) }, { x: new Date(2021, 1, 20) }, { x: new
Date(2021, 1, 21) }, { x: new Date(2021, 1, 22) }, { x: new Date(2021, 3, 1)
},
  { x: new Date(2021, 3, 2) },  { x: new Date(2021, 4, 1) },  { x: new
Date(2021, 4, 5) },  { x: new Date(2021, 4, 6) },  { x: new Date(2021, 4, 7)
},
  { x: new Date(2021, 4, 11) }, { x: new Date(2021, 4, 13) }, { x: new
Date(2021, 4, 15) }, { x: new Date(2021, 4, 16) }, { x: new Date(2021, 4,
17) },
  { x: new Date(2021, 4, 18) }, { x: new Date(2021, 4, 20) }, { x: new
Date(2021, 4, 21) }, { x: new Date(2021, 4, 23) }, { x: new Date(2021, 4,
25) },
  { x: new Date(2021, 5, 1) },  { x: new Date(2021, 5, 2) },  { x: new
Date(2021, 5, 6) },  { x: new Date(2021, 5, 7) },  { x: new Date(2021, 5, 8)
},
  { x: new Date(2021, 5, 11) }, { x: new Date(2021, 5, 15) }, { x: new
Date(2021, 5, 18) }, { x: new Date(2021, 5, 20) }, { x: new Date(2021, 5,
25) },
  { x: new Date(2021, 6, 1) },  { x: new Date(2021, 6, 2) },  { x: new
Date(2021, 6, 3) },  { x: new Date(2021, 6, 4) },  { x: new Date(2021, 6, 5)
},
  { x: new Date(2021, 6, 10) }, { x: new Date(2021, 6, 11) }, { x: new
Date(2021, 6, 12) }, { x: new Date(2021, 6, 13) }, { x: new Date(2021, 6,
15) },
  { x: new Date(2021, 6, 16) }, { x: new Date(2021, 6, 17) }, { x: new
Date(2021, 6, 18) }, { x: new Date(2021, 6, 19) }, { x: new Date(2021, 6,
20) }
];
let series1 = [];
let point1;
for (var j = 1; j < 46; j++) {
  point1 = {
```

```
    x: datetimeCategoryData[j].x,
    y: getRandomInRange(120, 130),
    high: getRandomInRange(88, 92),
    low: getRandomInRange(76, 86),
    open: getRandomInRange(75, 85),
    close: getRandomInRange(85, 90),
    volume: getRandomInRange(660187068, 965935749),
  };
  series1.push(point1);
}
function getRandomInRange(min, max) {
  const randomDecimal = Math.random();
  const randomValue = randomDecimal * (max - min) + min;
  return randomValue;
}
export default {
  data() {
    return {
        seriesData: series1,
        primaryXAxis: {
          valueType: "DateTimeCategory",
          majorGridLines: { width: 0 },
          crosshairTooltip: { enable: true }
        },
        crosshair: {
         enable: true
        },
        tooltip: { enable: true, header: 'AAPL Stock Price' }
    };
  },
  provide: {
    stockChart: [
      DateTimeCategory, RangeTooltip, LineSeries, SplineSeries,
CandleSeries, Tooltip, Crosshair, HiloOpenCloseSeries, HiloSeries,
RangeAreaSeries, Trendlines, EmaIndicator, RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export
    ]
  }
};
</script>
<style>
  #container {
    height: 350px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/axis-types-cs3" %}

Note: To use this axis, we need to inject DateTimeCategoryService into the provide and set the valueType of axis to DateTimeCategory.

Logarithmic axis

<!-- markdownlint-disable MD033 -->

Logarithmic axis uses logarithmic scale and it is very useful in visualizing data, when it has numerical values in both lower order of magnitude (eg: 10⁻⁶) and higher order of magnitude (eg: 10⁶). To use Logarithmic axis, set the valueType of axis to Logarithmic.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' yName='volume' low='low' high='high'
open='open' close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries, Logarithmic,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
             valueType: 'Logarithmic',
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price'
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,Logarithmic,
```

```
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/axis-types-cs2" %}

Note: To use this axis, we need to inject LogarithmicService into the provide and set the valueType of axis to Logarithmic.

See also

- Axis Customization

Axis customization in Vue Stock chart component

Axis Crossing

An axis can be positioned in the Stock Chart area using crossesAt properties. The crossesAt property specifies the values (datetime, numeric, or logarithmic) at which the axis line has to be intersected with the vertical axis or vice-versa, and the crossesInAxis property specifies the axis name with which the axis line has to be crossed.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
            crossesAt : 15
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 },
            crossesAt : 5
        },
      title: 'AAPL Stock Price'
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/axis-customization-cs1" %}

Title

You can add a title to the axis using title property to provide quick information to the user about the data plotted in the axis. Title style can be customized using titleStyle property of the axis.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
```

```
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
           title: 'AAPL Historical'
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price'
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/axis-customization-cs2" %}

Copyright © 2001 -2024 Syncfusion Inc.

Tick Lines Customization

You can customize the width, color and size of the minor and major tick lines, using majorTickLines and minorTickLines properties in the axis.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
              majorTickLines : {
                color : 'green',
                width : 5
            },
            minorTickLines : {
                color : 'red',
                width : 0
            }
       },
        primaryYAxis: {
             majorTickLines : {
                color : 'blue',
                width : 5
            },
            minorTickLines : {
                color : 'red',
```

```
                width : 0
            }
        },
      title: 'AAPL Stock Price'
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/axis-customization-cs3" %}

Grid Lines Customization

You can customize the `width`, `color` and `dashArray` of the minor and major grid lines, using majorGridLines and minorGridLines properties in the axis.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
              majorGridLines : {
              color : 'blue',
              width : 1
           },
           minorGridLines : {
              color : 'red',
              width : 0
           }
       },
        primaryYAxis: {
            majorGridLines : {
              color : 'green',
              width : 1
           },
           minorGridLines : {
              color : 'red',
              width : 0
           }
        },
      title: 'AAPL Stock Price'
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/axis-customization-cs4" %}

Copyright © 2001 -2024 Syncfusion Inc.

Multiple Axis

In addition to primary X and Y axis, we can add n number of axis to the stock chart. Series can be associated with this axis, by mapping with axis's unique name.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :axes='axes'
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
           <e-stockchart-series :dataSource="seriesData" type="Line"
xName='date' yName='close' yAxisName='yAxis'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" }
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
         axes:
          [
            {
            majorGridLines: { width: 0 },
            rowIndex: 1, opposedPosition: true,
            name: 'yAxis'
            }
        ],
```

```
      title: 'AAPL Stock Price'
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/axis-customization-cs5" %}

Inversed Axis

<!-- markdownlint-disable MD033 -->

When an axis is inversed, highest value of the axis comes closer to origin and vice versa. To place an axis in inversed manner set this property isInversed to true.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
            isInversed: true
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 },
            isInversed: true
        },
      title: 'AAPL Stock Price'
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/axis-customization-cs6" %}

Opposed Position

<!-- markdownlint-disable MD012 -->

To place an axis opposite from its original position, set opposedPosition property of the axis to true.

<!-- markdownlint-disable MD012 -->

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" }
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 },
            opposedPosition: true
        },
      title: 'AAPL Stock Price'
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/axis-customization-cs7" %}

Copyright © 2001 -2024 Syncfusion Inc.

Series types in Vue Stock chart component

Essential JS 2 StockChart supports 6 major types of series namely Line, Spline, Hilo, HiloOpenClose, Hollow Candle and Candle . By using the series dropdown button you can navigate between the above listed series types.

<!-- markdownlint-disable MD036 -->

Line

To render a line series, use series type as Line and

inject LineSeries into the provide.

Spline

To render a spline series, use series type as Spline and inject SplineSeries into the provide.

Hilo

To render a hilo series, use series type as Hilo and

inject HiloSeries into the provide.

HiloOpenClose

To render a hiloOpenClose series, use series type as HiloOpenClose and inject HiloOpenCloseSeries into the provide.

HollowCandle

To render a hollowcandle series, use series type as Candle and set enableSolidCandle as false. Now inject CandleSeries into the provide.

Candle

To render a candle series, use series type as Candle and inject CandleSeries into the provide.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :indicatorType="indicatorType"
        :trendlineType="trendlineType"
        :exportType="exportType"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      indicatorType:[],
      trendlineType: [],
      exportType: [],
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/series-types-cs1" %}

<!-- markdownlint-disable MD036 -->

Trend lines in Vue Stock chart component

Trendlines are used to show the direction and speed of price.

StockChart supports 6 types of trendlines namely Linear,Exponential,Logarithmic,Polynomial,Power,Moving Average. By using trendline dropdown button you can add or remove the required trendline type.

Linear

A linear trendline is a best fit straight line that is used with simpler data sets. To render a linear trendline, use trendline type as Linear and inject

Trendlines module using provide.

Copyright © 2001 -2024 Syncfusion Inc.

Exponential

An exponential trendline is a curved line that is most useful when data values rise or fall at increasingly higher rates. You cannot create an exponential trendline, if your data contains zero or negative values.

To render a exponential trendline, use trendline type as Exponential and inject Trendlines module using provide.

Logarithmic

A logarithmic trendline is a best-fit curved line that is most useful when the rate of change in the data increases or decreases quickly and then levels out. A logarithmic trendline can use negative and/or positive values.

To render a logarithmic trendline, use trendline type as Logarithmic and inject Trendlines module using provide.

Polynomial

A polynomial trendline is a curved line that is used when data fluctuates.

To render a polynomial trendline, use trendline type as Polynomial and inject Trendlines module using provide.

polynomialOrder used to define the polynomial value.

Power

A power trendline is a curved line that is best used with data sets that compare measurements that increase at a specific rate.

To render a power trendline, use trendline type as Power and inject Trendlines module using provide.

Moving Average

A moving average trendline smoothen out fluctuations in data to show a pattern or trend more clearly.

To render a moving average trendline, use trendline type as MovingAverage and inject Trendlines module using provide.

period property defines the period to find the moving average.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :seriesType="seriesType"
        :indicatorType="indicatorType"
        :exportType="exportType"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'>
          <e-trendlines>
              <e-trendline :type='type'></e-trendline>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          </e-trendlines>
          </e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,Trendlines
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
         seriesType:[],
        indicatorType:[],
        exportType: [],
        type: 'MovingAverage',
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip,  CandleSeries, Trendlines
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/trend-lines-cs1" %}

Customization of Trendlines

The fill and width properties are used to customize the appearance of the trendline.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
```

```
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :seriesType="seriesType"
        :indicatorType="indicatorType"
        :exportType="exportType"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'>
          <e-trendlines>
              <e-trendline :type='type'></e-trendline>
          </e-trendlines>
          </e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,Trendlines
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
         seriesType:[],
        indicatorType:[],
        exportType: [],
        type: 'MovingAverage',
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip,  CandleSeries, Trendlines
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/stockchart/trend-lines-cs2" %}

<!-- markdownlint-disable MD036 -->

Technical indicators in Vue Stock chart component

A technical indicator is a mathematical calculation based on historic price, volume or open interest information that aims to forecast financial market direction.

StockChart supports 10 types of technical indicators namely Accumulation Distribution, ATR, EMA,SMA,TMA,Momentum,MACD,RSI,Stochastic,Bollinger Band. By using indicator dropdown box you can add an remove the required indicators types.

Accumulation Distribution

Accumulation Distribution combines price and volume to show how money may be flowing into or out of stock.

To render a Accumulation Distribution Indicator, use indicator type as AccumulationDistribution and inject AccumulationDistributionIndicator into the provide. To calculate the signal line volume field is additionally added with dataSource.

Average True Range (ATR)

ATR measures the stock volatility by comparing the current value with the previous value. To render a Average True Range (ATR) Indicator, use indicator type as Atr and inject AtrIndicator into the provide.

Exponential Moving Average (EMA)

Moving average Indicators are used to define the direction of the trend. To render a EMA Indicator,

use indicator type as Ema and inject EmaIndicator into the provide.

Momentum

Momentum shows the speed at which the price of the stock is changing. To render a Momentum indicator, use indicator type as Momentumand inject MomentumIndicator into the provide. Momentum indicator will be represented by two lines (upperLine,

signalLine).In momentum indicator the upperBand value is always renders at the value 100.

Moving Average Convergence Divergence (MACD)

MACD is based on the difference between two EMA's. To render a MACD Indicator, use indicator type as Macd and inject MacdIndicator into the provide.MACD indicator will be represented by MACD line,signal line, MACD histogram. MACD histogram is used to differentiate MACD line and signal line.

Relative Strength Index (RSI)

RSI shows how strongly a stock is moving in its current direction. To render a RSI Indicator, use indicator type as Rsi and inject RsiIndicator into the provide.RSI indicator will be represented by three lines (upperBand, lowerBand, signalLine). The upperBand and lowerBand values are customized by overBought and overSold properties of indicator and the signalLine is calculated by RSI formula.

Simple Moving Average (SMA)

Moving average Indicators are used to define the direction of the trend. To render a SMA Indicator, use indicator type as Sma and inject SmaIndicator into the provide.

Copyright © 2001 -2024 Syncfusion Inc.

Stochastic

It shows how a stock is, when compared to previous state. To render a Stochastic indicator, use indicator type as Stochastic and inject StochasticIndicator into the provide. stochastic indicator will be represented by four lines (upperLine, lowerLine, periodLine, signalLine). In stochastic indicator the upperBand value and lowerBand value is customized by overBought and overSold properties of indicators and the periodLine and signalLine is render based on stochastic formula.

Triangular Moving Average (TMA)

Moving average Indicators are used to define the direction of the trend. To render a TMA Indicator, use indicator type as Tma and inject TmaIndicator into the provide.

Bollinger Band

A StockChart overlay that shows the upper and lower limits of normal price movements based on the standard deviation of prices. To render a Bollinger Band, use indicator type as BollingerBand and inject BollingerBands into the provide.

Bollinger band will be represented by three lines (upperLine, lowerLine, signalLine).The default values of the Bollinger Band period is 14 and standardDeviations is 2.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :seriesType="seriesType"
        :trendlineType="trendlineType"
        :exportType="exportType"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip, RangeAreaSeries,
Logarithmic, Zoom,
    EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator,
MomentumIndicator, SmaIndicator, AtrIndicator,
AccumulationDistributionIndicator, MacdIndicator, StochasticIndicator
} from "@syncfusion/ej2-vue-charts"
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
```

```
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      seriesType:[],
      trendlineType: [],
      exportType: [],
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, CandleSeries, EmaIndicator,
RsiIndicator,RangeAreaSeries,Logarithmic, Zoom,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/technical-indicators-cs1" %}

Legend in Vue Stock chart component

Legend provides information about the series rendered in the Stock Chart which can be customized using legend properties. Legend can be added to a Stock Chart by enabling the visible option in the legendSettings.

Position and Alignment

By using the position property, legend can be placed at Left, Right, Top, Bottom or Custom of the Stock Chart. The legend is positioned at the bottom of the Stock Chart, by default.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :tooltip="tooltip"
        :legendSettings="legendSettings"
        :indicatorType="indicatorType"
        :trendlineType="trendlineType"
        :crosshair="crosshair"
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open' close='close'
name='AAPL'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,Tooltip,
  Crosshair,AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export, StockLegend
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        trendlineType:[],
        indicatorType:[],
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
        crosshair: {
            enable: true,
        },
        tooltip: { enable: true },
        legendSettings: {
        visible: true,
        //Legend position as top
        position:'Top'
        },
        title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator, Tooltip,
  Crosshair,BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export, StockLegend
    ]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/legend-cs1" %}

- Custom position is used to position the legend anywhere in the Stock Chart using x, y coordinates.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :tooltip="tooltip"
        :legendSettings="legendSettings"
        :indicatorType="indicatorType"
        :trendlineType="trendlineType"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open' close='close'
name='AAPL'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,Tooltip,
  Crosshair,AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export, StockLegend
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
```

```
        trendlineType:[],
        indicatorType:[],
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
        crosshair: {
            enable: true,
        },
        tooltip: { enable: true },
        legendSettings: {
        visible: true,
        //Legend position as custom
        position:'Custom',
        location: { x: 200, y: 20 } },
        title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator, Tooltip,
  Crosshair,BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export, StockLegend
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/legend-cs2" %}

<!-- markdownlint-disable MD036 -->

Legend Alignment

<!-- markdownlint-disable MD036 -->

The legend can be align as `Center`, `Far` or `Near` to the chart using alignment property.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        :primaryYAxis="primaryYAxis"
        :tooltip="tooltip"
        :legendSettings="legendSettings"
        :indicatorType="indicatorType"
        :trendlineType="trendlineType"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open' close='close'
name='AAPL'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,Tooltip,
  Crosshair,AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export, StockLegend
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        trendlineType:[],
        indicatorType:[],
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
        crosshair: {
            enable: true,
        },
        tooltip: { enable: true },
        legendSettings: {
        visible: true,
        position:'Bottom',
        //Legend alignment as near
        alignment: 'Near'
        },
        title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
```

```
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator, Tooltip,
  Crosshair,BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export, StockLegend
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/legend-cs3" %}

Customization

To change the legend icon shape, legendShape property in the series can be used. By default legend icon shape is seriesType.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :tooltip="tooltip"
        :legendSettings="legendSettings"
        :indicatorType="indicatorType"
        :trendlineType="trendlineType"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open' close='close'
name='AAPL' legendShape='Pentagon'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,Tooltip,
```

```
  Crosshair,AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export, StockLegend
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        trendlineType:[],
        indicatorType:[],
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
        crosshair: {
            enable: true,
        },
        tooltip: { enable: true },
        legendSettings: { visible: true },
        title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator, Tooltip,
  Crosshair,BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export, StockLegend
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/legend-cs4" %}

Legend Size

By default, legend takes 20% - 25% of the Stock Chart's height horizontally, when it is placed on top or bottom position and 20% - 25% of the Stock Chart's width vertically, when placed on left or right position of the Stock Chart. The default legend size can be changed by using the width and height property of the legendSettings.

APP.VUE

```
<template>
  <div class="control-section">
```

```
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :tooltip="tooltip"
        :legendSettings="legendSettings"
        :indicatorType="indicatorType"
        :trendlineType="trendlineType"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open' close='close'
name='AAPL'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,Tooltip,
  Crosshair,AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export, StockLegend
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        trendlineType:[],
        indicatorType:[],
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
        crosshair: {
            enable: true,
        },
        tooltip: { enable: true },
        legendSettings: {
        visible: true,
        //Legend size for chart
        width: '500', height: '50',
        border: { width: 1, color: 'pink'}
        },
        title: 'AAPL Stock Price',
```

```
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator, Tooltip,
  Crosshair,BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export, StockLegend
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/legend-cs5" %}

Legend Item Size

The size of the legend items can customized by using the shapeHeight and shapeWidth property.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :tooltip="tooltip"
        :legendSettings="legendSettings"
        :indicatorType="indicatorType"
        :trendlineType="trendlineType"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open' close='close'
name='AAPL'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,Tooltip,
  Crosshair,AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export, StockLegend
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        trendlineType:[],
        indicatorType:[],
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
        crosshair: {
            enable: true,
        },
        tooltip: { enable: true },
        legendSettings: {
        visible: true,
        //Legend item size for chart
        shapeHeight: 15, shapeWidth: 15
        },
        title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator, Tooltip,
  Crosshair,BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export, StockLegend
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/legend-cs6" %}

Collapsing Legend Item

By default, series name will be displayed as legend. To skip the legend for a particular series, empty string to the series name can be given.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :tooltip="tooltip"
        :legendSettings="legendSettings"
        :indicatorType="indicatorType"
        :trendlineType="trendlineType"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,Tooltip,
  Crosshair,AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export, StockLegend
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        trendlineType:[],
        indicatorType:[],
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
        crosshair: {
            enable: true,
        },
        tooltip: { enable: true },
        title: 'AAPL Stock Price',
        legendSettings: { visible: true }
    };
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator, Tooltip,
  Crosshair,BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export, StockLegend
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/legend-cs7" %}

Legend Title

The title for legend can be set using title property in legendSettings. Customize the fontStyle, size, fontWeight, color, textAlignment, fontFamily, opacity and textOverflow of legend title. titlePosition is used to set the legend position in Top, Left and Right position. maximumTitleWidth is used to set the width of the legend title. By default, it will be 100px.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :tooltip="tooltip"
        :legendSettings="legendSettings"
        :indicatorType="indicatorType"
        :trendlineType="trendlineType"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open' close='close'
name='AAPL'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
```

```
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,Tooltip,
  Crosshair,AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export, StockLegend
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        trendlineType:[],
        indicatorType:[],
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
        crosshair: {
            enable: true,
        },
        tooltip: { enable: true },
        title: 'AAPL Stock Price',
        legendSettings: {
            visible: true,
            title: 'Countries',
            titlePosition: 'Top',
            titleStyle: {
                fontFamily: 'verdana',
                fontStyle: 'Normal',
                fontWeight: 'Normal',
                size: '15px',
                textAlignment: 'Center',
                color: 'blue',
                textOverflow: 'None'
            },
            maximumTitleWidth: 150
        }
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator, Tooltip,
  Crosshair,BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export, StockLegend
    ]
  }
};
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/legend-cs8" %}

Note: To use legend feature, we need to inject StockLegend into the Provide.

Tool tip in Vue Stock chart component

<!-- markdownlint-disable MD036 -->

StockChart will display details about the points through tooltip, when the mouse is moved over the point.

Default tooltip

By default, tooltip is not visible. Enable the tooltip by setting enable property to true and by injecting Tooltip

into the provide.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :tooltip="tooltip"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,Tooltip,
  Crosshair,AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
```

```
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
       crosshair: {
        enable: true,
      },
      tooltip: { enable: true },
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator, Tooltip,
  Crosshair,BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/tool-tip-cs1" %}

Format the tooltip

<!-- markdownlint-disable MD013 -->

By default, tooltip shows information of x and y value in points. In addition to that, you can show more information in tooltip. For example the format '${series.name} ${point.x}' shows series name and point x value.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :tooltip="tooltip"
        :crosshair="crosshair"
```

```
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,Tooltip,
  Crosshair,AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
       crosshair: {
        enable: true,
      },
      tooltip: { enable: true, header: 'Stock Price',
             format: '<b>${point.x} : ${point.y}</b>' },
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator, Tooltip,
  Crosshair,BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/tool-tip-cs2" %}

Position the tooltip

By default, the tooltip is positioned at the left side of the stock chart. You can move the tooltip along with the mouse by setting **Nearest** to the position property.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :tooltip="tooltip"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,Tooltip,
  Crosshair,AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
       crosshair: {
        enable: true,
```

```
      },
      tooltip: { enable: true, shared: true, position: 'Nearest' },
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator, Tooltip,
  Crosshair,BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/tool-tip-cs4" %}

Customize the appearance of the tooltip

The fill and border properties are used to customize the background color and border of the tooltip respectively. The textStyle property in the tooltip is used to customize the font of the tooltip text.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :tooltip="tooltip"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,Tooltip,
  Crosshair,AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
       crosshair: {
        enable: true,
      },
      tooltip: { enable: true,
                 fill: '#7bb4eb',
                 border: {
                    width: 2,
                    color: 'grey'
       } },
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator, Tooltip,
  Crosshair,BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/tool-tip-cs3" %}

Cross hair in Vue Stock chart component

Crosshair has a vertical and horizontal line to view the value of the axis at mouse or touch position.

Crosshair lines can be enabled by using enableproperty in the crosshair.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        width='650px' height='350px'
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, Crosshair,AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
        crosshair: {
         enable: true,
        },
       title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, Crosshair, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,     HiloOpenCloseSeries, HiloSeries, RangeAreaSeries,
Trendlines, EmaIndicator, RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/cross-hair-cs1" %}

Tooltip for axis

Tooltip label for an axis can be enabled by using enable property of crosshairTooltip in the corresponding axis.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, Crosshair,AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
            crosshairTooltip: { enable: true }
        },
```

```
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 },
            crosshairTooltip: { enable: true }
        },
        crosshair: {
         enable: true,
        },
       title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, Crosshair, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,     HiloOpenCloseSeries, HiloSeries, RangeAreaSeries,
Trendlines, EmaIndicator, RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/cross-hair-cs2" %}

Customization

The fill and textStyle property of the crosshairTooltip is used to customize the background color and font style of the crosshair label respectively. Color and width of the crosshair line can be customized by using the line property in the crosshair.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :crosshair="crosshair"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, Crosshair,AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
            crosshairTooltip: { enable: true,fill: 'green' }
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 },
            crosshairTooltip: { enable: true ,fill: 'green'}
        },
        crosshair: {  enable: true, line: {width: 2, color: 'green'}, fill:
'green' }
       title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, Crosshair, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,     HiloOpenCloseSeries, HiloSeries, RangeAreaSeries,
Trendlines, EmaIndicator, RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/cross-hair-cs3" %}

Note: To use crosshair feature, we need to inject Crosshair into the provide.

Period selector in Vue Stock chart component

The period selector allows to select a range with specified periods. By default the period selector is enabled in stock chart.

Copyright © 2001 -2024 Syncfusion Inc.

Periods

Periods is an array of objects that allows users to specify the range of periods. The interval property specifies the count value of the button, and the text property specifies the text to be displayed on button. The intervalType property allows users to customize the intervals of the buttons. The intervalType property supports the following interval types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-periods>
          <e-stockchart-period intervalType="Minutes" interval="1"
text="1m"></e-stockchart-period>
          <e-stockchart-period intervalType="Minutes" interval="30"
text="30m"></e-stockchart-period>
          <e-stockchart-period intervalType="Hours" interval="1"
text="1H"></e-stockchart-period>
          <e-stockchart-period intervalType="Hours" interval="12" text="12H"
selected="true"></e-stockchart-period>
          <e-stockchart-period text="1D"></e-stockchart-period>
        </e-stockchart-periods>
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/period-selector-cs1" %}

Visibility of period selector

The enablePeriodSelector property allows users to toggle the visibility of period selector.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :enablePeriodSelector="enablePeriodSelector"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
```

```
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
        enablePeriodSelector: false,
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/period-selector-cs2" %}

Copyright © 2001 -2024 Syncfusion Inc.

Range selector in Vue Stock chart component

The left and right thumb of RangeNavigator are used to indicate the selected range in the large collection of data. Following are the ways you can select a range.

- By dragging the thumbs.
- By tapping on the labels.
- By setting the start and end through Date Range button.

Following code example shows the enableSelector property allows users to toggle the visibility of enable selector.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :enableSelector="enableSelector"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
        enableSelector: false,
```

```
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/range-selector-cs1" %}

Export print in Vue Stock chart component

The rendered stock chart can be exported to JPEG, PNG, SVG, or PDF format using the export dropdown button in the period selector toolbar. You can choose the required format using the export dropdown button in stock-chart.

The rendered stock chart can be printed directly using print button in period selector toolbar.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/export-print-cs1" %}

Disable Export and print

To empty the value of exportType for to disable the Export button.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :exportType="exportType"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, RangeTooltip,
LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        exportType: [],
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price',
    };
  },
  provide: {
    stockChart: [
      DateTime, RangeTooltip, LineSeries, SplineSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines, EmaIndicator,
RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator, AccumulationDistributionIndicator, MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/export-print-cs2" %}

Copyright © 2001 -2024 Syncfusion Inc.

Appearance in Vue Stock chart component

Stock Chart Title

Chart can be given a title using title property, to show the information about the data plotted.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, Tooltip, Crosshair,
RangeTooltip, LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price'
    };
  },
  provide: {
    stockChart: [
      DateTime, Tooltip, Crosshair, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,     HiloOpenCloseSeries, HiloSeries, RangeAreaSeries,
Trendlines, EmaIndicator, RsiIndicator,
```

```
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/appearance-cs1" %}

Title Customization

The textStyle property of chart title provides options to customize the size, color, fontFamily, fontWeight, fontStyle, opacity, textAlignment and textOverflow.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title"
        :titleStyle="titleStyle">
        <e-stockchart-series-collection>
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, Tooltip, Crosshair,
RangeTooltip, LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
```

```
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price',
      titleStyle:{
            fontFamily: "Arial",
            fontStyle: 'italic',
            color: "#E27F2D",
            size: '23px'
        }
    };
  },
  provide: {
    stockChart: [
      DateTime, Tooltip, Crosshair, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,     HiloOpenCloseSeries, HiloSeries, RangeAreaSeries,
Trendlines, EmaIndicator, RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/appearance-cs2" %}

Stock Chart Theme

Changing theme will affect background color,gridlines,tooltip colors and appearance.

theme property of Stock chart is shipped with several built-in themes such as Material, Fabric, Bootstrap , HighContrastLight, MaterialDark, FabricDark, FabricDark, HighContrast and BootstrapDark.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :title="title"
        :theme="theme">
        <e-stockchart-series-collection>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          <e-stockchart-series :dataSource="seriesData" type="Candle"
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
        </e-stockchart-series-collection>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
  StockChartPlugin, DateTime, CandleSeries, Tooltip, Crosshair,
RangeTooltip, LineSeries,SplineSeries,
  HiloOpenCloseSeries, HiloSeries, RangeAreaSeries, Trendlines,
EmaIndicator, RsiIndicator,BollingerBands,  TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator, AccumulationDistributionIndicator,
MacdIndicator, StochasticIndicator, Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
        seriesData:chartData,
        primaryXAxis: {
            valueType: "DateTime",
            majorGridLines: { color: "transparent" },
        },
        primaryYAxis: {
            majorTickLines: { color: "transparent", width: 0 }
        },
      title: 'AAPL Stock Price',
      theme:'HighContrast'
    };
  },
  provide: {
    stockChart: [
      DateTime, Tooltip, Crosshair, RangeTooltip, LineSeries, SplineSeries,
CandleSeries,     HiloOpenCloseSeries, HiloSeries, RangeAreaSeries,
Trendlines, EmaIndicator, RsiIndicator,
      BollingerBands, TmaIndicator, MomentumIndicator, SmaIndicator,
AtrIndicator,      AccumulationDistributionIndicator,  MacdIndicator,
StochasticIndicator,Export
    ]
  }
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/appearance-cs3" %}

Copyright © 2001 -2024 Syncfusion Inc.

See Also

- Axis Customization

<!-- markdownlint-disable MD036 -->

Stock events in Vue Stock chart component

Stock Events visualizes stock events in stock chart. 'SplineSeries' is used to represent selected data value. You can customize the specific data value using stockEvents event.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        style="display:block"
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :chartArea="chartArea"
        :crosshair="crosshair"
        :tooltip="tooltip"
        :seriesType="seriesType"
        :title="title"
        :indicatorType="indicator"
        :border="border"
      >
        <e-stockchart-series-collection>
          <e-stockchart-series
            :dataSource="seriesData"
            type="Spline"
            xName="date"
            name="Apple Inc"
            yName="high"
            close="high"
          ></e-stockchart-series>
        </e-stockchart-series-collection>
         <e-stockchart-stockevents>
          <e-stockchart-stockevent :date="date1" text="Q2" description="2012
Quarter2 starts"
            type="Flag" ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date2" text="Open"
description="Markets opened"
            :textStyle="textStyle" background="#f48a21"
:border="borderCircle" ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date3" text="Q3" description="2013
Quarter3 starts"
            type="Flag" :textStyle="textStyle" background="#6c6d6d"
:border="borderFlag">
            </e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date4" text="Q4" description="2013
Quarter4 starts"
            type="Flag" :textStyle="textStyle" background="#6c6d6d"
:border="borderFlag">
             </e-stockchart-stockevent>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          <e-stockchart-stockevent :date="date5" text="G"
description="Google stocks bought"
            :textStyle="textStyle" background="#f48a21"
:border="borderCircle" >
            </e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date6" text="Y" description="Yahoo
stocks sold"
            type="Square" :textStyle="textStyle" background="#841391"
:border="borderSquare">
            </e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date7" text="Y2" description="Year
2013" type="Pin"
            :showOnSeries="onAxis" :textStyle="textStyle"
background="#6322e0" :border="borderPin" ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date8" text="Q2" description="2013
Quarter2 starts"
            type="Flag" :textStyle="textStyle" background="#6c6d6d"
:border="borderFlag">
             </e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date9" text="Q2"
description="Surge in Stocks"     type="ArrowUp" :textStyle="textStyle"
background="#3ab0f9" :border="borderArrow"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date10" text="Q3"
description="2013 Quarter3 starts"
            type="Flag" :textStyle="textStyle" background="#6c6c6d"
:border="borderFlag"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date11" text="Q4"
description="2013 Quarter4 starts"
            type="Flag" :textStyle="textStyle" background="#6c6c6d"
:border="borderFlag"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date12" text="Y3"
description="Year 2014"
            type="Pin" :showOnSeries="onAxis" :textStyle="textStyle"
background="#6322e0"
            :border="borderPin" ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date13" text="Q2"
description="2014 Quarter2 starts"
            type="ArrowDown" :textStyle="textStyle" background="#3ab0f9"
:border="borderArrow"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date14" text="Q3"
description="2014 Quarter3 starts"
            :textStyle="textStyle" background="#f48a21"
:border="borderCircle" >
            </e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date15" text="Q4"
description="2014 Quarter4 starts"
            type="Flag" :textStyle="textStyle" background="#6c6d6d"
:border="borderFlag"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date16" text="Y4"
description="Year 2015" type="Pin"
            :showOnSeries="onAxis" :textStyle="textStyle"
background="#6322e0"
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            :border="borderPin" ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date17" text="End"
description="Markets closed"
            type="ArrowDown" :textStyle="textStyle" background="#3ab0f9"
:border="borderArrow"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date18" text="A" description="This
is event description"
            :textStyle="textStyle" background="#f48a21"
:border="borderCircle" ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date19" text="Q1" description="Add
longer text"
            :textStyle="textStyle" background="#dd3c9f" :border="borderText"
type="Text"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date20" text="Close"
description="Markets closed"
            :textStyle="textStyle" background="#f48a21"
:border="borderCircle"
          ></e-stockchart-stockevent>
        </e-stockchart-stockevents>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
   StockChartPlugin, DateTime, SplineSeries, Crosshair, Tooltip,
RangeTooltip, LineSeries,
  CandleSeries, HiloOpenCloseSeries, HiloSeries, RangeAreaSeries,
Trendlines, EmaIndicator,
  RsiIndicator, BollingerBands, TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,
  AccumulationDistributionIndicator, MacdIndicator, StochasticIndicator,
Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
      seriesData: chartData,
      seriesType: [],
      indicator: [],
      primaryXAxis: {
        valueType: "DateTime",
        majorGridLines: { color: "transparent" },
        crosshairTooltip: { enable: true }
      },
      primaryYAxis: {
        lineStyle: { color: "transparent" },
        majorTickLines: { color: "transparent" },
        crosshairTooltip: { enable: true }
      },
      chartArea: {
        border: {
```

```
          width: 0
        }
      },
      date1: new Date(2012, 3, 1),
      date2: new Date(2012, 3, 20),
      date3: new Date(2012, 6, 1),
      date4: new Date(2012, 9, 1),
      date5: new Date(2012, 7, 30),
      date6: new Date(2012, 10, 1),
      date7: new Date(2012, 12, 0),
      date8: new Date(2013, 3, 1),
      date9: new Date(2013, 3, 20),
      date10: new Date(2013, 6, 1),
      date11: new Date(2013, 9, 1),
      date12: new Date(2013, 12, 0),
      date13: new Date(2014, 3, 1),
      date14: new Date(2014, 6, 1),
      date15: new Date(2014, 9, 1),
      date16: new Date(2014, 12, 0),
      date17: new Date(2014, 2, 2),
      date18: new Date(2015, 1, 7),
      date19: new Date(2015, 1, 2),
      date20: new Date(2015, 2, 12),
      textStyle: { color: "white" },
      borderFlag: { color: '#6c6d6d'},
      borderSquare: { color: '#841391'},
      borderArrow: { color: '#3ab0f9'},
      borderPin: { color: '#6233e0'},
      borderText: { color: '#dd3c9f'},
      borderCircle: { color: '#f48a21'},
      title: "AAPL Stock Price",
      tooltip: { enable: true },
      crosshair: { enable: true },
      border: { width: 0 },
      onAxis: false
    };
  },
  provide: {
    stockChart: [
      DateTime, SplineSeries, Crosshair,Tooltip, RangeTooltip, LineSeries,
  CandleSeries, HiloOpenCloseSeries, HiloSeries, RangeAreaSeries,
Trendlines, EmaIndicator,
  RsiIndicator, BollingerBands, TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,
  AccumulationDistributionIndicator, MacdIndicator, StochasticIndicator,
Export
    ]
  },
  methods: {}
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/stock-events-cs1" %}

Stock Events for individual series

By default, stock events will be showed for all series. Now, you can set the stock events for particular series using seriesIndexes property.

APP.VUE

```
<template>
  <div class="control-section">
    <div>
      <ejs-stockchart
        style="display:block"
        id="stockchartcontainer"
        :primaryXAxis="primaryXAxis"
        :primaryYAxis="primaryYAxis"
        :chartArea="chartArea"
        :crosshair="crosshair"
        :tooltip="tooltip"
        :seriesType="seriesType"
        :title="title"
        :indicatorType="indicator"
        :border="border"
      >
        <e-stockchart-series-collection>
          <e-stockchart-series
            :dataSource="seriesData"
            type="Spline"
            xName="date"
            yName="high"
          ></e-stockchart-series>
          <e-stockchart-series
            :dataSource="seriesData"
            type="Spline"
            xName="date"
            yName="low"
          ></e-stockchart-series>
          <e-stockchart-series
            :dataSource="seriesData"
            type="Spline"
            xName="date"
            yName="open"
          ></e-stockchart-series>
        </e-stockchart-series-collection>
         <e-stockchart-stockevents>
          <e-stockchart-stockevent :date="date1" text="Q2" description="2012
Quarter2 starts"
            type="Flag" :seriesIndexes="index1"></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date2" text="Open"
description="Markets opened"
            :textStyle="textStyle" background="#f48a21"
:border="borderCircle" ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date3" text="Q3" description="2013
Quarter3 starts"
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            type="Flag" :textStyle="textStyle" background="#6c6d6d"
:border="borderFlag" :seriesIndexes="index2">
            </e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date4" text="Q4" description="2013
Quarter4 starts"
            type="Flag" :textStyle="textStyle" background="#6c6d6d"
:border="borderFlag" :seriesIndexes="index3">
             </e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date5" text="G"
description="Google stocks bought"
            :textStyle="textStyle" background="#f48a21"
:border="borderCircle" >
            </e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date6" text="Y" description="Yahoo
stocks sold"
            type="Square" :textStyle="textStyle" background="#841391"
:border="borderSquare" :seriesIndexes="index4">
            </e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date7" text="Y2" description="Year
2013" type="Pin"
            :showOnSeries="onAxis" :textStyle="textStyle"
background="#6322e0" :border="borderPin" ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date8" text="Q2" description="2013
Quarter2 starts"
            type="Flag" :textStyle="textStyle" background="#6c6d6d"
:border="borderFlag" :seriesIndexes="index5">
             </e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date9" text="Q2"
description="Surge in Stocks"     type="ArrowUp" :textStyle="textStyle"
background="#3ab0f9" :border="borderArrow" :seriesIndexes="index6"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date10" text="Q3"
description="2013 Quarter3 starts"
            type="Flag" :textStyle="textStyle" background="#6c6c6d"
:border="borderFlag"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date11" text="Q4"
description="2013 Quarter4 starts"
            type="Flag" :textStyle="textStyle" background="#6c6c6d"
:border="borderFlag"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date12" text="Y3"
description="Year 2014"
            type="Pin" :showOnSeries="onAxis" :textStyle="textStyle"
background="#6322e0"
            :border="borderPin" ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date13" text="Q2"
description="2014 Quarter2 starts"
            type="ArrowDown" :textStyle="textStyle" background="#3ab0f9"
:border="borderArrow" :seriesIndexes="index7"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date14" text="Q3"
description="2014 Quarter3 starts"
            :textStyle="textStyle" background="#f48a21"
:border="borderCircle" >
            </e-stockchart-stockevent>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          <e-stockchart-stockevent :date="date15" text="Q4"
description="2014 Quarter4 starts"
            type="Flag" :textStyle="textStyle" background="#6c6d6d"
:border="borderFlag" :seriesIndexes="index8"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date16" text="Y4"
description="Year 2015" type="Pin"
            :showOnSeries="onAxis" :textStyle="textStyle"
background="#6322e0"
            :border="borderPin" ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date17" text="End"
description="Markets closed"
            type="ArrowDown" :textStyle="textStyle" background="#3ab0f9"
:border="borderArrow" :seriesIndexes="index9"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date18" text="A" description="This
is event description"
            :textStyle="textStyle" background="#f48a21"
:border="borderCircle" ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date19" text="Q1" description="Add
longer text"
            :textStyle="textStyle" background="#dd3c9f" :border="borderText"
type="Text" :seriesIndexes="index1"
          ></e-stockchart-stockevent>
          <e-stockchart-stockevent :date="date20" text="Close"
description="Markets closed"
            :textStyle="textStyle" background="#f48a21"
:border="borderCircle" :seriesIndexes="index2"
          ></e-stockchart-stockevent>
        </e-stockchart-stockevents>
      </ejs-stockchart>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { chartData } from "./datasource.js";
import {
   StockChartPlugin, DateTime, SplineSeries, Crosshair, Tooltip,
RangeTooltip, LineSeries,
  CandleSeries, HiloOpenCloseSeries, HiloSeries, RangeAreaSeries,
Trendlines, EmaIndicator,
  RsiIndicator, BollingerBands, TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,
  AccumulationDistributionIndicator, MacdIndicator, StochasticIndicator,
Export
} from "@syncfusion/ej2-vue-charts";
Vue.use(StockChartPlugin);
export default {
  data() {
    return {
      seriesData: chartData,
      seriesType: [],
      indicator: [],
      primaryXAxis: {
        valueType: "DateTime",
        majorGridLines: { color: "transparent" },
```

```
        crosshairTooltip: { enable: true }
      },
      primaryYAxis: {
        lineStyle: { color: "transparent" },
        majorTickLines: { color: "transparent" },
        crosshairTooltip: { enable: true }
      },
      chartArea: {
        border: {
          width: 0
        }
      },
      date1: new Date(2012, 3, 1),
      date2: new Date(2012, 3, 20),
      date3: new Date(2012, 6, 1),
      date4: new Date(2012, 9, 1),
      date5: new Date(2012, 7, 30),
      date6: new Date(2012, 10, 1),
      date7: new Date(2012, 12, 0),
      date8: new Date(2013, 3, 1),
      date9: new Date(2013, 3, 20),
      date10: new Date(2013, 6, 1),
      date11: new Date(2013, 9, 1),
      date12: new Date(2013, 12, 0),
      date13: new Date(2014, 3, 1),
      date14: new Date(2014, 6, 1),
      date15: new Date(2014, 9, 1),
      date16: new Date(2014, 12, 0),
      date17: new Date(2014, 2, 2),
      date18: new Date(2015, 1, 7),
      date19: new Date(2015, 1, 2),
      date20: new Date(2015, 2, 12),
      textStyle: { color: "white" },
      borderFlag: { color: '#6c6d6d'},
      borderSquare: { color: '#841391'},
      borderArrow: { color: '#3ab0f9'},
      borderPin: { color: '#6233e0'},
      borderText: { color: '#dd3c9f'},
      borderCircle: { color: '#f48a21'},
      title: "AAPL Stock Price",
      tooltip: { enable: true },
      crosshair: { enable: true },
      border: { width: 0 },
      onAxis: false,
      index1: [0],
      index2: [1],
      index3: [2],
      index4: [0],
      index5: [1],
      index6: [2],
      index7: [0],
      index8: [1],
      index9: [2]
    };
  },
  provide: {
    stockChart: [
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      DateTime, SplineSeries, Crosshair,Tooltip, RangeTooltip, LineSeries,
  CandleSeries, HiloOpenCloseSeries, HiloSeries, RangeAreaSeries,
Trendlines, EmaIndicator,
  RsiIndicator, BollingerBands, TmaIndicator, MomentumIndicator,
SmaIndicator, AtrIndicator,
  AccumulationDistributionIndicator, MacdIndicator, StochasticIndicator,
Export
    ]
  },
  methods: {}
};
</script>
<style>
#container {
   height: 350px;
 }
</style>
```

{% previewsample "page.domainurl/code-snippet/stockchart/stock-events-cs2" %}

Accessibility in Vue Stock chart component

The Stock chart component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Stock chart component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/not-supported.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Stock chart component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Stock chart component:

- img (role)
- button (role)
- region (role)
- aria-label (attribute)
- aria-hidden (attribute)
- aria-pressed (attribute)

Keyboard interaction

The Stock chart component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Stock chart component.

| **Press** | **To do this** |

| --- | --- |

| Alt + J | Moves the focus to the Stock chart element. |

| Tab | Moves the focus to the next element in the Stock chart. |

| Shift + Tab | Moves the focus to the previous element in the Stock chart. |

| Down Arrow | Moves the focus to the data point left side from the selected point. |

| Up Arrow | Moves the focus to the data point right side from the selected point. |

| ESC | Cancel the tooltip for the data point. |

| Ctrl + P | Prints the Stock chart. |

Copyright © 2001 -2024 Syncfusion Inc.

Ensuring accessibility

The Stock chart component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Stock chart component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Stock chart component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/stock-chart.html" %}

See also

- Accessibility in Syncfusion Vue components

Switch

Getting Started with the Vue Switch Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Switch component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The list of dependencies required to use the Switch component in your application is given below:

`js

|-- @syncfusion/ej2-vue-buttons

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-buttons

|-- @syncfusion/ej2-vue-base

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

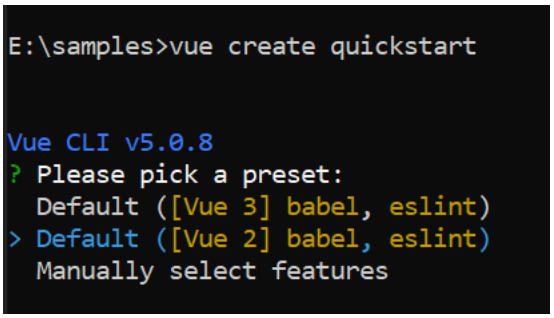
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the [Vue Switch component]https://www.syncfusion.com/vue-components/vue-toggle-switch-button) as an example. Install the @syncfusion/ej2-vue-buttons package by running the following command:

`bash

npm install @syncfusion/ej2-vue-buttons --save

`

or

`bash

yarn add @syncfusion/ej2-vue-buttons

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Switch component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Switch component using Composition API or Options API:

1\. First, import and register the Switch component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SwitchComponent as EjsSwitch } from "@syncfusion/ej2-vue-buttons";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SwitchComponent } from "@syncfusion/ej2-vue-buttons";
export default {
components: {
'ejs-switch': SwitchComponent
}
}
</script>
```

2\. In the template section, define the Switch component with the checked property.

~/SRC/APP.VUE

```
<template>
<ejs-switch checked=true></ejs-switch>
</template
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <ejs-switch checked=true></ejs-switch>
</template>
<script setup>
import { SwitchComponent as EjsSwitch } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.e-switch-wrapper {
  margin: 18px;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <ejs-switch checked=true></ejs-switch>
</template>
<script>
import { SwitchComponent } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
export default {
  components: {
    'ejs-switch': SwitchComponent
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.e-switch-wrapper {
  margin: 18px;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/switch/getting-started-cs2" %}

Set text on Switch

Switch component has text support. This can be achieved by setting onLabel

and offLabel on Switch.

Switch does not have text support for material themes, and does not support long custom text.

Getting Started with the Vue Switch Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Switch component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

Copyright © 2001 -2024 Syncfusion Inc.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

Copyright © 2001 -2024 Syncfusion Inc.

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Switch component as an example. To use the Vue Switch component in the project, the @syncfusion/ej2-vue-buttons package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-buttons --save

`

or

`bash

yarn add @syncfusion/ej2-vue-buttons

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

Copyright © 2001 -2024 Syncfusion Inc.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Switch component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue Switch component using Composition API or Options API:

1.First, import and register the Switch component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { SwitchComponent as EjsSwitch } from "@syncfusion/ej2-vue-buttons";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { SwitchComponent } from "@syncfusion/ej2-vue-buttons";
//Component registration
export default {
name: "App",
components: {
"ejs-switch": SwitchComponent
}
}
</script>
```

2.In the template section, define the Switch component with the checked property.

~/SRC/APP.VUE

```
<template>
<ejs-switch checked="isChecked"></ejs-switch>
</template>
```

3.Declare the values for the checked property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
bool isChecked = true;
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
isChecked: true,
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-switch checked="isChecked"></ejs-switch>
</template>
<script setup>
import { SwitchComponent as EjsSwitch } from "@syncfusion/ej2-vue-buttons";
bool isChecked = true;
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-switch checked="isChecked"></ejs-switch>
</template>
<script>
import { SwitchComponent } from "@syncfusion/ej2-vue-buttons";
// Component registration
export default {
name: "App",
// Declaring component and its directives
components: {
"ejs-switch": SwitchComponent
},
// Bound properties declarations
data() {
return {
isChecked: true,
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

![vue-3-js-switch](images/vue-3-js-switch.png)

Sample: vue-3-switch-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Accessibility in Vue Switch component

The Switch component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Switch component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Switch component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Switch component:

| Attributes | Purpose |

| --- | --- |

| role | Indicates the switch component. |

| aria-disabled | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Switch component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Switch component.

| **Press** | **To do this** |

| --- | --- |

| Space | When the switch has focus, pressing the Space key changes the state of the switch. |

Ensuring accessibility

The Switch component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

Copyright © 2001 -2024 Syncfusion Inc.

The accessibility compliance of the Switch component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Switch component with accessibility tools.

{% previewsample "page.domainurl/code-snippet/switch/getting-started-cs1" %}

See also

- Accessibility in Syncfusion Vue components

Two way binding in Vue Switch component

It can be achieved by using the v-model directive in vue. In the following sample, when you enable or disable a value in one Switch will automatically enable or disable the value in other Switch. It updates the other Switch using value property.

APP.VUE

```
<template>
<div>
<div id="wrapper1">
<ejs-switch v-model="value" :checked="value"></ejs-switch>
</div>
<div id="wrapper2">
<ejs-switch v-model="value" :checked="value"></ejs-switch>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { SwitchPlugin } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SwitchPlugin);
export default {
  data(){
    return{
      value: true
    }
  }
}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
.e-switch-wrapper {
  margin: 18px;
}
#wrapper1 {
  float: left;
  padding: 0 0 0 80px;
}
#wrapper2 {
  float: right;
  padding: 0 460px 0 0;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/radio-button/default-cs11" %}

Style and appearance in Vue Switch component

To modify the Switch appearance, you need to override the default CSS of Switch component. Please find the list of CSS classes and its corresponding section in Switch. Also, you have an option to create your own custom theme for the controls using our Theme Studio.

CSS Class | Purpose of Class

|.e-switch-wrapper .e-switch-inner|To customize the line of the switch in off mode

|.e-switch-wrapper .e-switch-handle|To customize the handle of the switch in off mode

|.e-switch-wrapper:not(.e-switch-disabled):hover .e-switch-handle:not(.e-switch-active)|To customize the handle of the switch in off mode when hover

|.e-switch-wrapper:not(.e-switch-disabled):hover .e-switch-inner:not(.e-switch-active)|To customize the line of the switch in off mode when hover

|.e-switch-wrapper .e-switch-handle.e-switch-active|To customize the handle of the switch in on mode

|.e-switch-wrapper .e-switch-on|To customize the line of the switch in on mode

|.e-switch-wrapper:hover .e-switch-handle.e-switch-active|To customize the handle of the switch in on mode when hover

|.e-switch-wrapper:hover .e-switch-inner.e-switch-active .e-switch-on|To customize the line of the switch in on mode when hover

How To

Change size in Vue Switch component

The different Switch sizes available are default and small. To reduce the size of default Switch to small, set the cssClass property to e-small.

APP.VUE

```
<template>
        <table class='size'>
            <tr>
                <td class='lSize'>Small</td>
                <td>
                    <ejs-switch id="switch1" checked=true cssClass="e-
small"></ejs-switch>
                </td>
            </tr>
            <tr>
                <td class='lSize'>Default</td>
                <td>
                    <ejs-switch id="switch2"></ejs-switch>
                </td>
            </tr>
        </table>
</template>
<script>
import Vue from 'vue';
import { SwitchPlugin } from "@syncfusion/ej2-vue-buttons";
```

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SwitchPlugin);
export default {}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/fabric.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/fabric.css";
.size tr td {
  padding: 10px;
}
.size .lSize {
  font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-
serif";
  font-size: 13px;
  cursor: pointer;
  user-select: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/switch/getting-started-cs3" %}

Enable ripple for switch label in Vue Switch component

By default, label with ripple effect is not available in Switch. You can achieve this using `rippleMouseHandler` method.

The following example illustrates how to enable ripple effect for labels in Switch component.

APP.VUE

```
<template>
    <table class='switch-control'>
        <tr>
            <td class='lSize'><label for='switch1'>USB
Tethering</label></td>
            <td>
                <ejs-switch id="switch1"></ejs-switch>
            </td>
        </tr>
    </table>
</template>
<script>
import Vue from 'vue';
import { SwitchPlugin } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
import { rippleMouseHandler } from '@syncfusion/ej2-buttons';
enableRipple(true);
Vue.use(SwitchPlugin);
export default {
    mounted: function(){
        var elemArray = document.querySelectorAll('.switch-control label');
        for (var i = 0, len = elemArray.length; i < len; i++) {
            elemArray[i].addEventListener('mouseup', rippleHandler);
            elemArray[i].addEventListener('mousedown', rippleHandler);
        }
        function rippleHandler(e) {
```

```
            var rippleSpan =
this.parentElement.nextElementSibling.querySelector('.e-ripple-container');
            rippleSpan && rippleMouseHandler(e, rippleSpan);
        }
    }
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/fabric.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/fabric.css";
.switch-control tr td {
  padding: 10px;
}
.switch-control .lSize {
  font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-
serif";
  font-size: 13px;
  cursor: pointer;
  user-select: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/switch/getting-started-cs5" %}

Enable rtl in Vue Switch component

Switch component has RTL support. This can be achieved by setting enableRtl as true.

The following example illustrates how to enable right-to-left support in Switch component.

APP.VUE

```
<template>
  <ejs-switch enableRtl=true></ejs-switch>
</template>
<script>
import Vue from 'vue';
import { SwitchPlugin } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SwitchPlugin);
export default {}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/fabric.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/fabric.css";
.e-switch-wrapper {
  margin: 18px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/switch/getting-started-cs6" %}

Set disabled state in Vue Switch component

Switch can be disabled by setting the disabled property to true.

The following example illustrates how to disable support in Switch component.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <ejs-switch disabled=true ></ejs-switch>
</template>
<script>
import Vue from 'vue';
import { SwitchPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(SwitchPlugin);
export default {}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/fabric.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/fabric.css";
.e-switch-wrapper {
  margin-top: 18px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/switch/getting-started-cs7" %}

Submit name and value in form in Vue Switch component

The name attribute of the Switch is used to group Switches. When the Switches are grouped in form, the checked items

value attribute will post to the server on form submit. The disabled and unchecked Switch values will not be sent to

the server on form submit.

In the following code snippet, USB and Wi-Fi in the checked state, and Bluetooth is in disabled state. Values that are in checked state only be sent on form submit.

APP.VUE

```
<template>
    <form>
        <table class='size'>
            <tr>
                <td class='lSize'><label>USB</label></td>
                <td>
                    <ejs-switch id="switch1" name="tethering" value="USB"
checked=true></ejs-switch>
                </td>
            </tr>
            <tr>
                <td class='lSize'><label>Wi-Fi</label></td>
                <td>
                    <ejs-switch id="switch2" name="hotspot" value="wifi"
checked=true></ejs-switch>
                </td>
            </tr>
            <tr>
                <td class='lSize'><label class="e-
disabled">Bluetooth</label></td>
                <td>
```

```
                    <ejs-switch id="switch3" name="tethering"
value="bluetooth" disabled=true></ejs-switch>
                </td>
            </tr>
            <tr>
                <td>
                    <ejs-button id="btn" isPrimary=true>Submit</ejs-button>
                </td>
            </tr>
        </table>
    </form>
</template>
<script>
import Vue from 'vue';
import { SwitchPlugin, ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SwitchPlugin);
Vue.use(ButtonPlugin);
export default {}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/fabric.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/fabric.css";
.size tr td {
  padding: 10px;
}
.size .lSize {
  font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-
serif";
  font-size: 13px;
  cursor: pointer;
  user-select: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/switch/getting-started-cs8" %}

Change switch state using toggle method in Vue Switch component

This section explains about how to toggle between the switch states using toggle method.

APP.VUE

```
<template>
  <ejs-switch ref="toggleSwitch" checked=true :created="created"></ejs-
switch>
</template>
<script>
import Vue from 'vue';
import { SwitchPlugin } from "@syncfusion/ej2-vue-buttons";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(SwitchPlugin);
export default {
  methods: {
    created: function(args) {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      this.$refs.toggleSwitch.toggle();
    }
  }
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/fabric.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/fabric.css";
.e-switch-wrapper {
  margin: 18px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/switch/getting-started-cs4" %}

Switch triggers change event on every state stage to perform custom operations.

Tab

Getting Started with the Vue Tab Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Tab component

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following is the list of dependencies required to use the Tab component in your application.

`javascript

|-- @syncfusion/ej2-vue-navigations

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-navigations

|-- @syncfusion/ej2-inputs

|-- @syncfusion/ej2-buttons

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

Copyright © 2001 -2024 Syncfusion Inc.

`

or

`bash

yarn global add @vue/cli

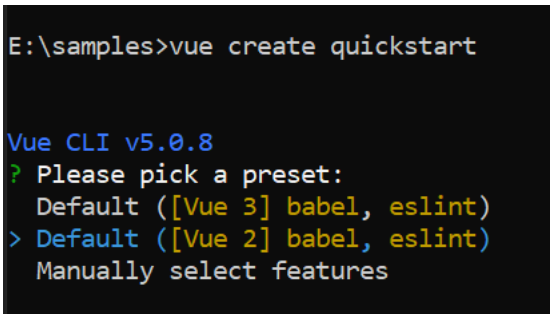
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Tab component as an example. Install the @syncfusion/ej2-vue-navigations package by running the following command:

`bash

npm install @syncfusion/ej2-vue-navigations --save

`

or

`bash

yarn add @syncfusion/ej2-vue-navigations

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

Copyright © 2001 -2024 Syncfusion Inc.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Tab component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Tab component:

1\. First, import and register the Tab component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { TabComponent, TabItemDirective, TabItemsDirective } from
'@syncfusion/ej2-vue-navigations';
export default {
components: {
'ejs-tab': TabComponent,
'e-tabitem': TabItemDirective,
'e-tabitems': TabItemsDirective
}
}
</script>
```

2\. In the template section, define the Tab component with content and header properties

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-tab id='element'>
<e-tabitems>
<e-tabitem :header='headerText0' :content="content0"></e-tabitem>
<e-tabitem :header='headerText1' :content="content1"></e-tabitem>
<e-tabitem :header='headerText2' :content="content2"></e-tabitem>
</e-tabitems>
</ejs-tab>
</div>
</template>
```

3\. Declare the value for the content and header properties in the script section.

~/SRC/APP.VUE

```
<script>
data: function(){
```

Copyright © 2001 -2024 Syncfusion Inc.

```
return {
headerText0: { text: 'ASP.NET' },
headerText1: { text: 'ASP.NET MVC' },
headerText2: { text: 'JavaScript' },
content0: 'ASP.NET is an open-source server-side web application framework
designed for web development to produce ' +
'dynamic web pages. It was developed by Microsoft to allow programmers to
build dynamic web sites, web applications ' +
'and web services. It was first released in January 2002 with version 1.0 of
the .NET Framework, and is the successor ' +
'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is built on
the Common Language Runtime (CLR), allowing ' +
'programmers to write ASP.NET code using any supported .NET language. The
ASP.NET SOAP extension framework allows ' +
'ASP.NET components to process SOAP messages.',
content1: 'The ASP.NET MVC is a web application framework developed by
Microsoft, which implements the ' +
'model-view-controller (MVC) pattern. It is open-source software, apart from
the ASP.NET Web Forms component which is ' +
'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web
API, and ASP.NET Web Pages (a platform using ' +
'only Razor pages) will merge into a unified MVC 6.The project is called
ASP.NET vNext.',
content2: 'JavaScript (JS) is an interpreted computer programming language.
It was originally implemented as ' +
'part of web browsers so that client-side scripts could interact with the
user, control the browser, communicate ' +
'asynchronously, and alter the document content that was displayed.[5] More
recently, however, it has become common in ' +
'both game development and the creation of desktop applications.',
}
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

~/SRC/APP.VUE

```
<template>
    <div id="app">
    <ejs-tab id='element'>
        <e-tabitems>
                    <e-tabitem :header='headerText0' :content="content0"></e-
tabitem>
                     <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                     <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
      </e-tabitems>
    </ejs-tab>
  </div>
</template>
<script>
import { TabComponent, TabItemDirective, TabItemsDirective } from
'@syncfusion/ej2-vue-navigations';
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  components: {
    'ejs-tab': TabComponent,
    'e-tabitem': TabItemDirective,
    'e-tabitems': TabItemsDirective
  },
  name: 'app',
   data: function(){
        return {
          headerText0: { text: 'ASP.NET' },
          headerText1: { text: 'ASP.NET MVC' },
          headerText2: { text: 'JavaScript' },
        content0: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
         content1: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content2: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
        }
   }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

Copyright © 2001 -2024 Syncfusion Inc.

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/tab/getting-started-cs1" %}

Initialize the Tab using HTML elements

The Tab component can be rendered based on the given HTML element using <ejs-tab> tag.

Header section must be enclosed with in a wrapper element using e-tab-header class and corresponding content must be mapped with e-content class.

You need to follow the below structure of HTML elements to render the Tab,

`

<ejs-tab id="element"> --> Root Tab element

 --> Tab header

 --> Header Item

 --> Tab content

 --> Content Item

</ejs-tab>

`

~/SRC/APP.VUE

```
<template>
    <div id="app">
    <ejs-tab id='element'>
         <div class="e-tab-header">
                <div>Twitter </div>
                <div>Facebook </div>
                <div>WhatsApp </div>
            </div>
            <div class="e-content">
                <div>
                        Twitter is an online social networking service that
enables users to send and read short 140-character messages called 'tweets'.
Registered users can read and post tweets, but those who are unregistered
can only read them. Users access Twitter through the website interface, SMS
or mobile device app Twitter Inc. is based in San Francisco and has more
than 25 offices around the world. Twitter was created in March 2006 by Jack
```

```
Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in July 2006.
The service rapidly gained worldwide popularity, with more than 100 million
users posting 340 million tweets a day in 2012.The service also handled 1.6
billion search queries per day.
                </div>
                <div>
                        Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was launched on
February 4, 2004, by Mark Zuckerberg with his Harvard College roommates and
fellow students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris
Hughes.The founders had initially limited the website's membership to
Harvard students, but later expanded it to colleges in the Boston area, the
Ivy League, and Stanford University. It gradually added support for students
at various other universities and later to high-school students.
                </div>
                <div>
                        WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates under a subscription
business model. It uses the Internet to send text messages, images, video,
user location and audio media messages to other users using standard
cellular mobile numbers. As of February 2016, WhatsApp had a user base of up
to one billion,[10] making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
by Facebook Inc. on February 19, 2014, for approximately US$19.3 billion.
                </div>
            </div>
    </ejs-tab>
  </div>
</template>
<script>
import { TabComponent } from '@syncfusion/ej2-vue-navigations';
export default {
  components: {
        'ejs-tab': TabComponent
  },
  name: 'app',
   data: function(){
        return {
        }
   }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/tab-container-cs1" %}

Initialize the Tab using V-slot Template

In the Vue Tab component, the v-slot directive is used to define a slot template in the component's template where users can insert custom content. Refer to the following code sample.

~/SRC/APP.VUE

```
<template>
        <div id="app">
                <ejs-tab id="element">
                        <e-tabitems>
                                <e-tabitem :header='datePickerText'
:content="'datePickerTemplate'">
                                        <template v-slot:datePickerTemplate>
                                                <ejs-datepicker></ejs-
datepicker>
                                        </template>
                                </e-tabitem>
                                <e-tabitem :header='calendarText'
:content="'calendarTemplate'">
                                        <template v-slot:calendarTemplate>
                                                <ejs-calendar></ejs-
calendar>
                                        </template>
                                </e-tabitem>
                        </e-tabitems>
                </ejs-tab>
        </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { DatePickerPlugin, CalendarPlugin } from '@syncfusion/ej2-vue-
calendars';
Vue.use(TabPlugin);
Vue.use(DatePickerPlugin);
Vue.use(CalendarPlugin);
export default {
        name: 'app',
        data: function () {
                return {
                        datePickerText: { text: 'Datepicker' },
                        calendarText: { text: 'Calendar' },                }
        }
}
</script>

<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/tab-v-slot-template" %}

See Also

- How to load tab with DataSource

Copyright © 2001 -2024 Syncfusion Inc.

Note: You can refer to our Vue Tab feature tour page for its groundbreaking feature representations. You can also explore our Vue Tab Component example that shows how to render the Tab in Vue.

Getting Started with Syncfusion Vue Tab Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Tab component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Setup the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will creates a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Copyright © 2001 -2024 Syncfusion Inc.

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion Vue Tab component to the project.

Adding Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Tab component as an example. To use the Vue Tab component in the project, the @syncfusion/ej2-vue-navigations package needs to be installed using the following command.

`bash

npm install @syncfusion/ej2-vue-navigations --save

`

Copyright © 2001 -2024 Syncfusion Inc.

or

`bash

yarn add @syncfusion/ej2-vue-navigations

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Tab component and its dependents were imported into the <style> section of the **src/App.vue** file.

`html

<style>

@import "../node_modules/@syncfusion/ej2-base/styles/material.css";

@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-navigations/styles/material.css";

</style>

`

Adding Syncfusion Vue Tab component in the application

Follow the below steps to add the Vue Tab component using Composition API:

1.First, import and register the Tab component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import {
TabComponent as EjsTab, TabItemsDirective as ETabitems, TabItemDirective as
ETabitem
} from "@syncfusion/ej2-vue-navigations";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import {
TabComponent,
TabItemDirective,
TabItemsDirective,
} from "@syncfusion/ej2-vue-navigations";
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
name: "App",
components: {
"ejs-tab": TabComponent,
"e-tabitem": TabItemDirective,
"e-tabitems": TabItemsDirective,
}
}
</script>
```

2.Add the component definition in template section.

`html

<template>

<ejs-tab id="tab">

<e-tabitems>

<e-tabitem :header="headerText0" :content="content0"></e-tabitem>

<e-tabitem :header="headerText1" :content="content1"></e-tabitem>

<e-tabitem :header="headerText2" :content="content2"></e-tabitem>

</e-tabitems>

</ejs-tab>

</template>

`

3.Declare the properties for Tab component.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const headerText0 = { text: "ASP.NET" };
const headerText1 = { text: "ASP.NET MVC" };
const headerText2 = { text: "JavaScript" };
const content0 =
"ASP.NET is an open-source server-side web application framework designed
for web development to produce " +
"dynamic web pages. It was developed by Microsoft to allow programmers to
build dynamic web sites, web applications " +
"and web services. It was first released in January 2002 with version 1.0 of
the .NET Framework, and is the successor " +
"to Microsoft's Active Server Pages (ASP) technology. ASP.NET is built on
the Common Language Runtime (CLR), allowing " +
"programmers to write ASP.NET code using any supported .NET language. The
ASP.NET SOAP extension framework allows " +
"ASP.NET components to process SOAP messages.";
const content1 =
"The ASP.NET MVC is a web application framework developed by Microsoft,
which implements the " +
"model-view-controller (MVC) pattern. It is open-source software, apart from
the ASP.NET Web Forms component which is " +
"proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web
API, and ASP.NET Web Pages (a platform using " +
```

Copyright © 2001 -2024 Syncfusion Inc.

```
"only Razor pages) will merge into a unified MVC 6.The project is called
ASP.NET vNext.";
const content2 =
"JavaScript (JS) is an interpreted computer programming language. It was
originally implemented as " +
"part of web browsers so that client-side scripts could interact with the
user, control the browser, communicate " +
"asynchronously, and alter the document content that was displayed.[5] More
recently, however, it has become common in " +
"both game development and the creation of desktop applications.";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data: function() {
return {
headerText0: { text: "ASP.NET" },
headerText1: { text: "ASP.NET MVC" },
headerText2: { text: "JavaScript" },
content0: "ASP.NET is an open-source server-side web application framework
designed for web development to produce " +
"dynamic web pages. It was developed by Microsoft to allow programmers to
build dynamic web sites, web applications " +
"and web services. It was first released in January 2002 with version 1.0 of
the .NET Framework, and is the successor " +
"to Microsoft's Active Server Pages (ASP) technology. ASP.NET is built on
the Common Language Runtime (CLR), allowing " +
"programmers to write ASP.NET code using any supported .NET language. The
ASP.NET SOAP extension framework allows " +
"ASP.NET components to process SOAP messages.",
content1: "The ASP.NET MVC is a web application framework developed by
Microsoft, which implements the " +
"model-view-controller (MVC) pattern. It is open-source software, apart from
the ASP.NET Web Forms component which is " +
"proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web
API, and ASP.NET Web Pages (a platform using " +
"only Razor pages) will merge into a unified MVC 6.The project is called
ASP.NET vNext.",
content2: "JavaScript (JS) is an interpreted computer programming language.
It was originally implemented as " +
"part of web browsers so that client-side scripts could interact with the
user, control the browser, communicate " +
"asynchronously, and alter the document content that was displayed.[5] More
recently, however, it has become common in " +
"both game development and the creation of desktop applications."
}
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-tab id="tab">
```

```
<e-tabitems>
<e-tabitem :header="headerText0" :content="content0"></e-tabitem>
<e-tabitem :header="headerText1" :content="content1"></e-tabitem>
<e-tabitem :header="headerText2" :content="content2"></e-tabitem>
</e-tabitems>
</ejs-tab>
</template>
<script setup>
import {
TabComponent as EjsTab, TabItemsDirective as ETabitems, TabItemDirective as
ETabitem
} from "@syncfusion/ej2-vue-navigations";
const headerText0 = { text: "ASP.NET" };
const headerText1 = { text: "ASP.NET MVC" };
const headerText2 = { text: "JavaScript" };
const content0 =
"ASP.NET is an open-source server-side web application framework designed
for web development to produce " +
"dynamic web pages. It was developed by Microsoft to allow programmers to
build dynamic web sites, web applications " +
"and web services. It was first released in January 2002 with version 1.0 of
the .NET Framework, and is the successor " +
"to Microsoft's Active Server Pages (ASP) technology. ASP.NET is built on
the Common Language Runtime (CLR), allowing " +
"programmers to write ASP.NET code using any supported .NET language. The
ASP.NET SOAP extension framework allows " +
"ASP.NET components to process SOAP messages.";
const content1 =
"The ASP.NET MVC is a web application framework developed by Microsoft,
which implements the " +
"model-view-controller (MVC) pattern. It is open-source software, apart from
the ASP.NET Web Forms component which is " +
"proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web
API, and ASP.NET Web Pages (a platform using " +
"only Razor pages) will merge into a unified MVC 6.The project is called
ASP.NET vNext.";
const content2 =
"JavaScript (JS) is an interpreted computer programming language. It was
originally implemented as " +
"part of web browsers so that client-side scripts could interact with the
user, control the browser, communicate " +
"asynchronously, and alter the document content that was displayed.[5] More
recently, however, it has become common in " +
"both game development and the creation of desktop applications.";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<ejs-tab id="tab">
<e-tabitems>
<e-tabitem :header="headerText0" :content="content0"></e-tabitem>
<e-tabitem :header="headerText1" :content="content1"></e-tabitem>
<e-tabitem :header="headerText2" :content="content2"></e-tabitem>
</e-tabitems>
</ejs-tab>
</template>
<script>
import {
TabComponent,
TabItemDirective,
TabItemsDirective,
} from "@syncfusion/ej2-vue-navigations";
export default {
name: "App",
components: {
"ejs-tab": TabComponent,
"e-tabitem": TabItemDirective,
"e-tabitems": TabItemsDirective,
},
data: function () {
return {
headerText0: { text: "ASP.NET" },
headerText1: { text: "ASP.NET MVC" },
headerText2: { text: "JavaScript" },
content0: "ASP.NET is an open-source server-side web application framework
designed for web development to produce " +
"dynamic web pages. It was developed by Microsoft to allow programmers to
build dynamic web sites, web applications " +
"and web services. It was first released in January 2002 with version 1.0 of
the .NET Framework, and is the successor " +
"to Microsoft's Active Server Pages (ASP) technology. ASP.NET is built on
the Common Language Runtime (CLR), allowing " +
"programmers to write ASP.NET code using any supported .NET language. The
ASP.NET SOAP extension framework allows " +
"ASP.NET components to process SOAP messages.",
content1: "The ASP.NET MVC is a web application framework developed by
Microsoft, which implements the " +
"model-view-controller (MVC) pattern. It is open-source software, apart from
the ASP.NET Web Forms component which is " +
"proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web
API, and ASP.NET Web Pages (a platform using " +
"only Razor pages) will merge into a unified MVC 6.The project is called
ASP.NET vNext.",
content2: "JavaScript (JS) is an interpreted computer programming language.
It was originally implemented as " +
"part of web browsers so that client-side scripts could interact with the
user, control the browser, communicate " +
"asynchronously, and alter the document content that was displayed.[5] More
recently, however, it has become common in " +
"both game development and the creation of desktop applications."
}
}
}
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

Refer the sample Vue 3 using Composition API Tab getting started

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Adaptive in Vue Tab component

The following section explains about rendering Tab when its width exceeds the viewable area or particularly in a given width. The available

modes are as follows:

- Scrollable
- Popup

Scrollable

The default overflow mode is Scrollable. Scrollable display mode supports displaying the Tab header items in a single line with horizontal

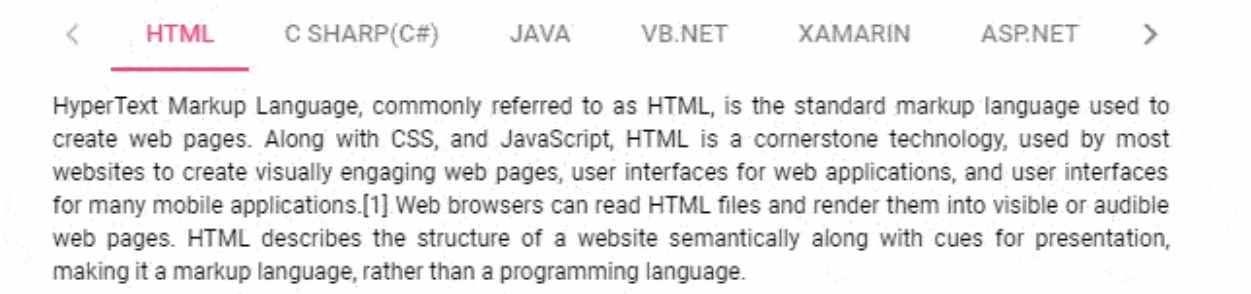
scrolling enabled, when the item overflows to the available space.

Copyright © 2001 -2024 Syncfusion Inc.

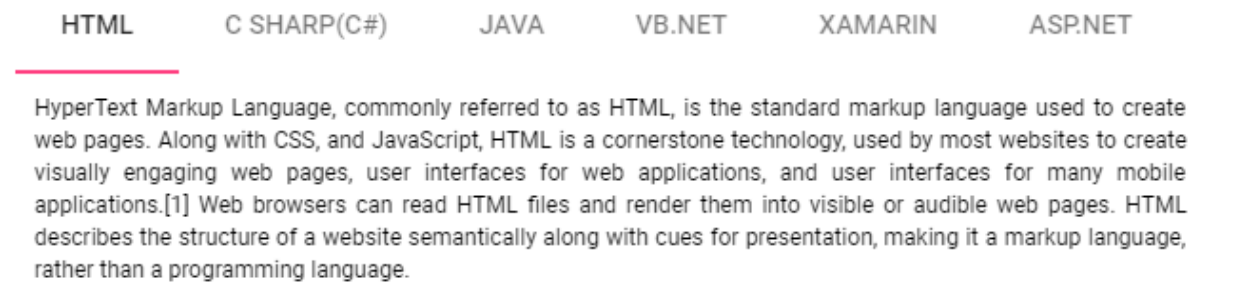
- The right and left navigation arrow is added at the start and end of the Tab header through which user can navigate towards overflowed

items of the Tab header.

- You can also see the overflowed items using touch and swipe action on the header and content section.
- By default, navigation icon in the left direction is disabled, you can see the overflowed items by moving in the right direction.
- By clicking the arrow or by holding the arrow continuously, you can see the overflowed items.



- In devices the navigation icons are not available. You can touch and swipe to see the overflowed items of the Tab header.



APP.VUE

```
<template>
    <div id="app">
    <ejs-tab id="element" heightAdjustMode='Auto' overflowMode='Scrollable'>
            <e-tabitems>
                 <e-tabitem :header='headerText0' :content="content0"></e-
tabitem>
                    <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                    <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
```

```
                    <e-tabitem :header='headerText3'
:content="content3"></e-tabitem>
                    <e-tabitem :header='headerText4'
:content="content4"></e-tabitem>
                    <e-tabitem :header='headerText5'
:content="content5"></e-tabitem>
                    <e-tabitem :header='headerText6'
:content="content6"></e-tabitem>
                    <e-tabitem :header='headerText7'
:content="content7"></e-tabitem>
            </e-tabitems>
        </ejs-tab>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(TabPlugin);
export default {
  name: 'app',
   data: function(){
        return {
          headerText0: { text: 'HTML' },
          headerText1: { text: 'C Sharp(C#)' },
          headerText2: { text: 'Java' },
          headerText3: { text: 'VB.Net' },
          headerText4: { text: 'Xamarin' },
          headerText5: { text: 'ASP.NET' },
          headerText6: { text: 'ASP.NET MVC' },
          headerText7: { text: 'JavaScript' },
          content0: 'HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to ' +
        'create web pages. Along with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually ' +
        'engaging web pages, user interfaces for web applications, and user
interfaces for many mobile applications.[1] Web ' +
        'browsers can read HTML files and render them into visible or
audible web pages. HTML describes the structure of a ' +
        'website semantically along with cues for presentation, making it a
markup language, rather than a programming language.',
         content1: 'C# is intended to be a simple, modern, general-purpose,
object-oriented programming language. Its ' +
        'development team is led by Anders Hejlsberg. The most recent
version is C# 5.0, which was released on August 15, 2012.',
        content2: 'Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired ' +
        'by Oracle Corporation, that provides a system for developing
application software and deploying it in a cross-platform ' +
        'computing environment. Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to ' +
        'enterprise servers and supercomputers. While less common, Java
applets run in secure, sandboxed environments to ' +
        'provide many features of native applications and can be embedded in
HTML pages.',
        content3: 'The command-line compiler, VBC.EXE, is installed as part
of the freeware .NET Framework SDK. Mono also ' +
```

```
        'includes a command-line VB.NET compiler. The most recent version is
VB 2012, which was released on August 15, 2012.',
        content4: 'Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers ' +
        'that created Mono,[4] Mono for Android and MonoTouch that are
cross-platform implementations of the Common Language ' +
        'Infrastructure (CLI) and Common Language Specifications (often
called Microsoft .NET). With a C#-shared codebase, ' +
        'developers can use Xamarin tools to write native Android, iOS, and
Windows apps with native user interfaces and share ' +
        'code across multiple platforms.[5] Xamarin has over 1 million
developers in more than 120 countries around the World ' +
        'as of May 2015.',
        content5: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
         content6: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content7: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
        }
   }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/adaptive/scrollable-cs1" %}

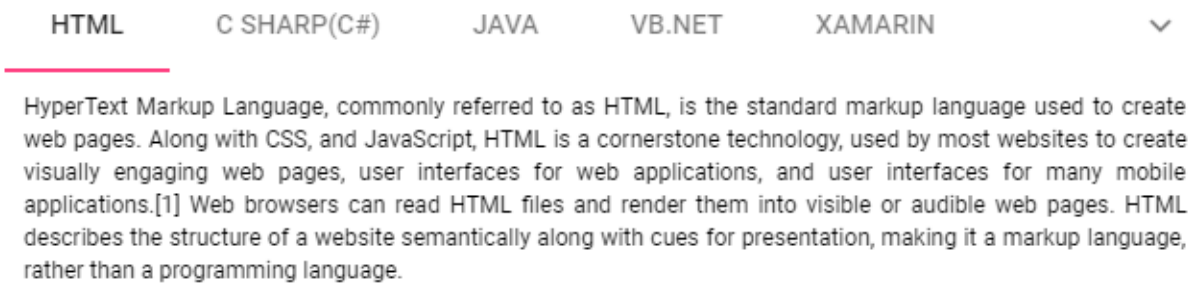
Copyright © 2001 -2024 Syncfusion Inc.

Popup

The Popup is the another type of overflowMode in which the Tab container holds the items that can be placed within the available space.

The rest of the overflowing items for which there is no space to fit within the viewing area are moved to overflow popup container.

- The items placed in popup can be viewed by opening the popup with the help of drop-down icon given at the end of the Tab header.
- If the popup height exceeds the height of the visible area, you can scroll through the popup items and select one.



APP.VUE

```
<template>
    <div id="app">
   <ejs-tab id="element" heightAdjustMode='Auto' overflowMode='Popup'>
     <e-tabitems>
                    <e-tabitem :header='headerText0'
:content="content0"></e-tabitem>
                    <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                    <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
                    <e-tabitem :header='headerText3'
:content="content3"></e-tabitem>
                    <e-tabitem :header='headerText4'
:content="content4"></e-tabitem>
                    <e-tabitem :header='headerText5'
:content="content5"></e-tabitem>
                    <e-tabitem :header='headerText6'
:content="content6"></e-tabitem>
                    <e-tabitem :header='headerText7'
:content="content7"></e-tabitem>
            </e-tabitems>
        </ejs-tab>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(TabPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  name: 'app',
     data: function(){
        return {
          headerText0: { text: 'HTML' },
          headerText1: { text: 'C Sharp(C#)' },
          headerText2: { text: 'Java' },
          headerText3: { text: 'VB.Net' },
          headerText4: { text: 'Xamarin' },
          headerText5: { text: 'ASP.NET' },
          headerText6: { text: 'ASP.NET MVC' },
          headerText7: { text: 'JavaScript' },
          content0: 'HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to ' +
        'create web pages. Along with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually ' +
        'engaging web pages, user interfaces for web applications, and user
interfaces for many mobile applications.[1] Web ' +
        'browsers can read HTML files and render them into visible or
audible web pages. HTML describes the structure of a ' +
        'website semantically along with cues for presentation, making it a
markup language, rather than a programming language.',
         content1: 'C# is intended to be a simple, modern, general-purpose,
object-oriented programming language. Its ' +
        'development team is led by Anders Hejlsberg. The most recent
version is C# 5.0, which was released on August 15, 2012.',
        content2: 'Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired ' +
        'by Oracle Corporation, that provides a system for developing
application software and deploying it in a cross-platform ' +
        'computing environment. Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to ' +
        'enterprise servers and supercomputers. While less common, Java
applets run in secure, sandboxed environments to ' +
        'provide many features of native applications and can be embedded in
HTML pages.',
        content3: 'The command-line compiler, VBC.EXE, is installed as part
of the freeware .NET Framework SDK. Mono also ' +
        'includes a command-line VB.NET compiler. The most recent version is
VB 2012, which was released on August 15, 2012.',
        content4: 'Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers ' +
        'that created Mono,[4] Mono for Android and MonoTouch that are
cross-platform implementations of the Common Language ' +
        'Infrastructure (CLI) and Common Language Specifications (often
called Microsoft .NET). With a C#-shared codebase, ' +
        'developers can use Xamarin tools to write native Android, iOS, and
Windows apps with native user interfaces and share ' +
        'code across multiple platforms.[5] Xamarin has over 1 million
developers in more than 120 countries around the World ' +
        'as of May 2015.',
        content5: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
         content6: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content7: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
        }
}
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/adaptive/popup-cs1" %}

See Also

- How to prevent content swipe selection
- Collapsible Tab

Header in Vue Tab component

This section explains about modifying the style of Tab header, and configuring its icons and positions.

Styles

You can customize header styles by adding predefined classes in the Tab root element. The pre-defined CSS class names are as follows:

- **e-fill**: The Selected Tab header background is set as solid fill.
- **e-background**: Tab header has a solid fill background, and the selected header has a highlighted border.
- **e-background e-accent**: Tab header has a solid fill background, and the selected header has a highlighted border with accent color.

Copyright © 2001 -2024 Syncfusion Inc.

If the above custom style classes are not included in the root element, the default style is applied to the Tab items.

APP.VUE

```
<template>
   <div id="wrapper" style='margin-top: 20px'>
   <div id='content' style="margin: 0px auto">
          <div id="default" style="padding-top:20px;width:250px">
            <div class='row'>
                <div>
                    <label> Header Style </label>
                </div><br/>
                <div>
                    <ejs-dropdownlist ref='headerStyles'
:dataSource='headerData' :change='onChange' :value='value' :fields='fields'
:popupHeight='height'></ejs-dropdownlist>
                </div>
            </div>
        </div>
        <br/>
     <div>
             <ejs-tab id='element' ref="element">
           <e-tabitems>
                    <e-tabitem :header='headerText0'
:content="content0"></e-tabitem>
                    <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                    <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
       </e-tabitems>
    </ejs-tab>
        </div>
      </div>
 </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(DropDownListPlugin);
Vue.use(TabPlugin);
export default {
  name: 'app',
     data: function(){
        return {
         headerData : [
        { Id: 'header1', headerStyle: 'default', text: 'Default' },
        { Id: 'header2', headerStyle: 'fill', text: 'Fill'},
        { Id: 'header3', headerStyle: 'background',text: 'Background' },
        { Id: 'header4', headerStyle: 'accent', text: 'Accent' }
        ],
      fields: { text: 'text', value: 'headerStyle' },
      height :'220px',
      value : 'default',
          headerText0: { text: 'ASP.NET' },
          headerText1: { text: 'ASP.NET MVC' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          headerText2: { text: 'JavaScript' },
         content0: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
         content1: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content2: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
        }
        },
        methods:{
      onChange: function(e){
       var tabObj =  this.$refs.element.ej2Instances;
        var listObj =  this.$refs.headerStyles.ej2Instances;
        this.removeStyleClass();
        if (listObj.value === 'fill') {
            tabObj.element.classList.add('e-fill');
        } else if (listObj.value === 'background') {
            tabObj.element.classList.add('e-background');
        } else if (listObj.value === 'accent') {
            tabObj.element.classList.add('e-background');
            tabObj.element.classList.add('e-accent');
        }
    }
      removeStyleClass: function(e) {
        var tabObj =  this.$refs.element.ej2Instances;
        tabObj.element.classList.remove('e-fill');
        tabObj.element.classList.remove('e-background');
        tabObj.element.classList.remove('e-accent');
    }
        }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/header-cs1" %}

Icon positions

You can customize the position of the Tab header icons using the iconPosition property. This property depends on the header items iconCss property. By default, Tab header icon is placed on left position. The position values are as follows:

- **Left**: Icon is placed on the left of the Tab header item.
- **Right**: Icon is placed on the right of the Tab header item.
- **Top**: Icon is placed on the top of the Tab header item.
- **Bottom**: Icon is placed on the bottom of the Tab header item.

APP.VUE

```
<template>
<div id="wrapper" style='margin-top: 20px'>
   <div id='content' style="margin: 0px auto">
          <div id="default" style="padding-top:20px;width:250px">
            <div class='row'>
                <div>
                    <label> Icon Position </label>
                </div><br/>
                <div>
                    <ejs-dropdownlist ref='iconPosition'
:dataSource='iconData' :change='onChange' :value='value' :fields='fields'
:popupHeight='height'></ejs-dropdownlist>
                </div>
            </div>
        </div>
        <br/>
     <div>
    <div id="app">
    <ejs-tab id='element' ref="element">
        <e-tabitems>
                  <e-tabitem :header='headerText0' :content="content0"></e-
tabitem>
                   <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                   <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
      </e-tabitems>
    </ejs-tab>
  </div>
</div>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(DropDownListPlugin);
Vue.use(TabPlugin);
export default {
  name: 'app',
   data: function(){
        return {
          iconData : [
        { Id: '1', position: 'left', text: 'Left' },
        { Id: '2', position: 'right', text: 'Right'},
        { Id: '3', position: 'top',text: 'Top' },
        { Id: '4', position: 'bottom', text: 'Bottom' }
        ],
      fields: { text: 'text', value: 'position' },
      height :'220px',
      value : 'left',
          headerText0: { 'text': 'Twitter', 'iconCss': 'e-twitter' },
          headerText1: { 'text': 'Facebook', 'iconCss': 'e-facebook' },
          headerText2: { 'text': 'WhatsApp', 'iconCss': 'e-whatsapp' },
        content0: 'Twitter is an online social networking service that
enables users to send and read short 140-character messages called
"tweets".' +
        'Registered users can read and post tweets, but those who are
unregistered can only read them. Users access Twitter' +
        'through the website interface, SMS or mobile device app Twitter
Inc. is based in San Francisco and has more than 25' +
        'offices around the world. Twitter was created in March 2006 by Jack
Dorsey, Evan Williams, Biz Stone, and Noah Glass' +
        'and launched in July 2006. The service rapidly gained worldwide
popularity, with more than 100 million users posting' +
        '340 million tweets a day in 2012.The service also handled 1.6
billion search queries per day.',
         content1: 'Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was launched on
February' +
        '4, 2004, by Mark Zuckerberg with his Harvard College roommates and
fellow students Eduardo Saverin, Andrew McCollum,' +
        'Dustin Moskovitz and Chris Hughes.The founders had initially
limited the website\'\s membership to Harvard students,' +
        'but later expanded it to colleges in the Boston area, the Ivy
League, and Stanford University. It gradually added support' +
        'for students at various other universities and later to high-school
students.',
         content2: 'WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates under a subscription'
+
         'business model. It uses the Internet to send text messages,
images, video, user location and audio media messages to' +
         'other users using standard cellular mobile numbers. As of February
2016, WhatsApp had a user base of up to one billion,[10]' +
         'making it the most globally popular messaging application.
WhatsApp Inc., based in Mountain View, California, was acquired' +
         'by Facebook Inc. on February 19, 2014, for approximately US$19.3
billion.',
        }
   },
   methods:{
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      onChange: function(e) {
        var tabObj =  this.$refs.element.ej2Instances;
        var listObj =  this.$refs.iconPosition.ej2Instances;
        console.log(tabObj);
        for(let i: number = 0; i < tabObj.items.length; i++) {
            tabObj.items[i].header.iconPosition = listObj.value;
        }
    }
}
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.e-content .e-item {
    font-size: 12px;
    margin: 10px;
    text-align: justify;
}
@font-face {
    font-family: 'Socialicons';
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMv1tCfsAAAEoAAAAVmNtYXCnKKeOAAABrAAAAEhnbHlml19
XagAAAgwAABhQaGVhZA8dCeEAAADQAAAANmhoZWEIUQQMAAAArAAAACRobXR4LAAAAAAAAYAAAAA
sbG9jYR3AIwwAAAH0AAAAGG1heHABIAIAAAABCAAAACBuYW1l0X1q/wAAGlwAAAJVcG9zdGX5D00
AABy0AAAAkwABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAACwABAAAAAQAA+iTiP18
PPPUACwQAAAAAANYFYngAAAAA1gVieAAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAALAfQACwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABApwCnCQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAAAACAAAAAwAAABQAAwABAAAAFAA
EADQAAAAEAAQAAQAApwn//wAApwD//wAAAAEABAAAAAEAAgADAAQABQAGAAcACAAJAAoAAAAAAiQ
CzgOMBU4F/gZYB9QIcAo+DCgABAAAAAAD0gPzAFUA4gF3AfMAAAEzHwYHFQ8EFR8IPwUfBRUPCCM
vFj0BPwoXNw8fHQEfDhUPAT8CHwkzPyA9Ai8iDwIFHwcPIysBLwYjDwI/AS8PNT8oHx4BDxAdAR8
PHQEHPwE7AR8EMz8dNS8kIw8FAYkFEgQDAyQDAQECAyIBAQMSEgkUCw4vBQQFChsGBQdqAgIBAwM
DCAoMDA0NBgYPEA8PFxYVFBQTEhITEREPDgwKCQQEBQICBAQFChMJBQUFBTURDxAPDw8ODg4NDQw
MDAsLCgkJCQgHBwcFBQUEAwICAQEDAgQEBgYHBwkJCgsOAgEmiwMEBAQUFRQVFRQVFRQVFRUVFBU
VDw4ODg0NDAwLCwoKCQkICAcGBgUEBAQCAgICAgMEBAYGBgcICQkJCgsLCwwNDQ0ODg4PDw8QEBA
QEBEREREQEQHcBgUEBAICAQEBAQEDAwQFBQYHCAgICgoLCwwNDQ4ODxAREhISEhMTExMUExQUFRQ
VGxsaGgcIBwfXNgEBAQ8KCgoIBwcGBQUDAwIBAQECAwMDBQUFBgcHCAgICgkKCwsMDAwNDg0ODw8
PEBAQEhISEhISEhIREREREREQERAQDw8PDw8ODQ0NDQwLDAoKCgkJCAcH/aAQEB0cGhgWFBIRDgw
LCAcEAwICAwMFBQYHBwgJCgoLAgE9+AYFBSMeHx4fIB8fFhQUFBQSExIRERAQEA8ODhAQDQ0LCQg
HBgQCAgICBAQEBgYGCAgJCQoLCwwMDQ4ODg8PEBAREBESEhISExITGBcZGRgZGBgXAu4CAgMEXAk
FBAQFBCQCAwMGGhcKFQkMIwIBAwoYAwIBKQECBSgFBgULCgkHBgMBAQEDAwcICgwMDg8PEhITFBU
WGBgSEyYJCAgIBwcHDBEGAgEBAagEBAQGBgYHCAgJCgkLCwsMDAwNDQ4ODg8PDw8QEBAQERITEhE
SERERERBEPEA8QDhMEBASFNgEBAQEJCAcGBQMCAQEDAwUGCAgHCAgJCQoKCwsMDA0NDQ4ODg8ODxA
PEA8QEBAREBAQEBIQERAQDw8ODg0NDQwMCwoKCQkICAcGBgUFAwMDAQEBAQEC6RISExISExISEhM
SEhESERERERAQEA8QDg8NDg0MDAwLCwoJCQcHBgUEAwICAgIEBggJAwECVNcFBAQVEBEREhESEhI
TExMTFBMUEhEREBEQEBAPDw8PDg4ODQ0MDAwLCwoKCgkICAcHBwUGBQQDAgIBAQEBAgIDBAQFBQc
GCAgICQoKCgsMDA0NDQ4PDw8QEBEBBwgIEhMUFhcYGRscHR8fIiIjFBMTEhMSEhIREhEREBEQEAM
EAwT1YQINCAYEAgIEBAUGBggICQoKDAwNDg8PERUVFhcWGBcYGBkZGRkaGxoTEhISEhEREBAQDw8
ODg4MDQwLCgoKCQgHBwYFBQMDAwEBAgMFBQgIAAAAAAEAAAAAAzoD9ACWAAATDwYVERUfHTsBPw4
9AS8OIy8PNSEzPw4vDyE9AS8ODwblCAYFBAQCAgECAwMEBQUGBwgICQoKCwwMDA0NDQ0ODg4PDw8
QEBDQCgsKCQkJCAgHBwUEBAICAgIEBAUHBwgICQkJCgsK0QoKCgoJCAgIBwcFBAMDAQEBKQoJCgg
JCAgHBwUFBAQCAQEBAQIEBAUFBwgHCAkJCQK/tcCAgMFBQYHCAkICQoJCwoLCwoJCQkIA9AJCgs
```

KDAwMDf4LExMTEhESEREQEBAPDw4PDQ4MDAoKCQgIBgYEBAMCAgEBAwQFBwcJCQoKCwsMDA0NDAs
MCwoKCQkHBwUEAwEBAQEDBAUGCAgJCgoLCwwMDVgCAwMFBgcICQkJCgsLCwwLDAsKCgoJCAgHBgU
EAgIBtQ0MDAsLCgoJCQcHBQQDAQEBAQMEBQYIAAABAAAAAAP0A90AqAAAAT8DMx8MHQEPDSsBLxo
PCBc/Ah8LEx8PMz8dLw8jDw4CSAoTEhIRCAcGBwUFBQQDAwICAgEDBQoPExYWFAsLCgQFBAUFBAU
FBQUJCQkJCyQEBQUGBwcICAkKCgsLDQwODQ8RERQfI5IvNRMGBwYGBwYGBgYGDAsMWAcICAgICQk
JCQkKCgoLCgsJExITFBQVFRYWFiMkJTEWFRQREQ8NDAsJBwYFAwEBAQEDBAUHBwkJCwwNDhAQEgs
YGBYWFBQSEhAQDg4MCwsC2wQGBQIBAgICAwQEBQUHBgcICBMSDxEdIiUoIxsOCgcCAQIEBAYICBU
bHyU37xQTERAPDQwKCQgGBQMCAQEDBggLDhofkUInCwIBAgQEBwcJCwscISf+pBYUExIQDg4LCwk
IBgUDAgEDBAcJDA0REhQXJysxRiEhIB4eHRwbGhkZFxcVFRoXFhQTERAODQwKCAcGBAMBAQMFBwk
MDQ8RExUXGhsdAAUAAAAAA/ED9ABCAKoA6wESAYQAAAEdAQ8NKwEvDjU/EB8OJR0BHw8hPw8TLwM
hHwUVDxEvEzU/CSchDwMFFR8PPw8vDw8OAR8HFQ8JIy8GPQI/BjMlHQEPBC8DNS8DDwMVDwIjLwM
9Ai8BIw8EFQ8DIy8CNw8KFxUfASUzPwgzHwkhPwI1LxAlDwICkAMDBQcHCQkKDAwMDQ4ODwwMCws
LCgoJDwsJCAYFAgECAwQGBgcICQkKCwsMDA0LCw8ODg4MDAsKCQkHBgQEAv1/AQMFBgkJDAwODwg
RERITEwJpExMSEhEQDw4MDAUJBwYEAgEBAgEF/uYOCwkGBAICBAYHCQsMDg8QERETExQTFRQVFBQ
UFBMTEhAPDg0LCQgGBAMCAgEBAwMEBQYHCAkC/uoFAgEBASwBAwUGCQoLDQ0PERESEhQUFBQTEhE
QEA4MDAkJBgUDAQEDBQcICgsNDg8QERISFBQUFBMSERAQDgwMCggHBAMCPAYGBgQEAwEBAQEBAwM
EBAQFBmgHBgYEBAMCAwMEBQUFBjX90AECBAUUBQEBAQEBAgIRAgIBAQIFEAkDAwECBAQEBQQDAwI
CAwUWAwIBAQQQDwwLCQgFAwEBAQEEATEEBBYUFRYWFxcYFxcXFxYVFBgFBQYBJwYCAgICBAYHCQo
LDA4ODxAIERIR/d8FAQIB9wcHDg0NDAwLCgkIBwYFBAICAgQEBQYGDQwODg8REBINDQwMCwsKCgk
IBwcGBAQCAQEBAgQFBggICgoLDQ0NDg9929sUExISERAPDgwMBQkHBgQCAQMFBgkJDAwODwgQERI
TEwHBBgMBARYXFxcXFxcWFhUUFBMREQ8ODAsJCAYEAwIBAgQFBwkKDA0PDxERExQPEA8PDw8PDw8
ODw4ODg4OAQEBAQKPCgoUEhIREBANDQsKCAcFAwEBAwUHCAoLDQ4PEBESExQUFBMTEhEQDw4NCwo
IBwUDAQEDBQcICgsNDg8QEhITEwGSAQICBAUFBgdsBQQFBAQDAgIBAQECAgQFBQYHawcHBgUDAgE
BR2h1CAMCAQEBAgIF5wMCAQEBAQED6gUCAQEDAwbbBQICAQIDAwMG0ggEAQICAgTKAQ0OEBASEhQ
VEiRdAgIBAQITDg0JCAYDAQQFBwoMDhQCAQEBAQNuJBIRERAPDg4NCwoJCAMFBAEBAQIEAAAAAAM
AAAAAA/QD3QADAFcAlwAANzMRIwUVIzc1IxEzET8OHw8RMxEvGw8MAR8PPw41Lw8PDhnW1gIjAQH
W1gIDBQgKCwcHBwgJCQoKCw4NDAsKCAgHBwUEBAICAQHWAQICAgQDBQUFBgYHBwcJCAkJCgoKCws
LDBgZGhQUEREPDg0MCwoJCQ79xAEBAwMFBgYHCAkKCwsMDA4PDQwLCwoJCQcGBgUDAwIBAQMEBAY
GCAgJCgoLDQwODQ0MDAoKCQkHBwYEBAMBIgKFWwICW/17AXcUDA0ODgwGBQUEBAMCAQEBAgMFBQc
ICgoLDA0NDw8Q/qcBhBIREBAPDw4NDQwMCwoKCQkICAcGBgUFBAMGAwEBAgMEBgYHCAgICQkSARI
MCwsKCgkICAgGBQUEAwEBAQEDBAUFBggICAkKCgsLDAsLCwsJCggIBwYGBAQDAQEBAQMEBAYGBwg
ICgkLCwsAAAABAAAAAAPuA/QARgAAExEVHwYhESM1MzU/DzMVIw8GFTMVIxEhPwYRLwYhDwYSAgQ
FBwgKCgHPb24BAwMGBggJCgsMDQ0ODwgPlUcLCwkIBgQDe3sBBQoKCAcFBAICBAUHCAoK/IUKCgk
HBwQDA7v8igYLCgkIBgQDAZuFUBAQDw4ODQwLCgkIBwUEAgGFAwQHCAkKDDOF/mUDBAYICQoLA4I
LCgkIBgQDAQQFBwgKCwAAAAAGAAAAAAP0A/QAOABEAIABBQEqAUwAAAEPCR0BHw07AT8NPQEvCCM
PASUVMxUjFSM1IzUzNSUPBRUfDTsBPww1Lw4jDwU3ByMfCA8PHw4dAQ8OLw8/DS8FPwIHIy8NPQE
/DwEVHw8hPw8RITchLw8hDw4BCgMTCwsFBAQEAgICAwQGBgcICgoLDAwODg8NDQwLCgkICAYGBQQ
DAwEBAQIDBAgMDiYRNw0B9nR0TXNz/kAFAwMDAQIBAgMDBAQGBgcICQkKDAsIBwcHBwYFBQYFAwM
BAQECAwMEBQYGBwgJCQoLDAcIBwcHBwX+MTAQDggIAwICAQEBAQEDAwMICgsMDAsGAgEBAQECAwY
iGQoFCQcDAgIBAwQFCAgLDA0PERITFRYYFRISEA8NDAsKCAcGBAMCAQEBAwUHCQsOERMUFB0xCAc
DAwEBAQIFGQ4ODQ0LCgoICAcFBQQCAgMDBgcICgwICBESEhESEBD+pwEDBQYJCgsNDg8IEBISExQ
CahQTExIREA8ODQsGCQcGBAL8GAED5gIDBgcICgsNDg4QCBIRExP9lhMTExEREA4ODQsKCAcGAwF
KAQkHCAYGBggICQkKCgkICAgHBgYFBQMDAgIBAgMDBAUFBgYHBwcICQgHBwYGBgYLCwwcBQPYck9
yck5zZwYGBwcHDxELCgwLCwsKCgkJBwUFAwECAwMDBAUHBwcIBw0QCwwLDAsMCgoKCAcGBAMBAgI
DAwQFLRkQDwwPCAgJCgoLCQkICAgNDAsKCQwJBQYGBQYEBAcbFQsGDA4HCAgJCQ4NDQwNCwwKCgg
IBgYDAwEBAgMDBQYGBwgJCAoJCgsKCwUMDAwMDAsKCQYFBQUKDAYHCAgJBw0BAgQEBQcHCAkJCgo
KCwsLDQ4NDQ0MDAsGBgkIBQQCAQH+EAoKExMSERAQDQ0LBgkHBgQCAQMFBgkKCw0NEAgQEhITFAJ
HKxQSEhIQDw8NDAsJBQcFBAIBAwQHCAkLDA0PDxASEhIAAAAAAgAAAAD7gP0AEAAAhAAAARUzFSM
RHws/BxUPAy8OESM1Pw8lER8OMyEzPw4RLw4jISMPDQIbysoDBgUICgYHCAgJCgsLDQ4PEBESE0Q
tICIiEREQDw8ODQwKCgcHBANuGBkVDw4ODgYFBgUEBAMCAv5fAQECAwQEBQUGBwcHCAgJCAM0CAk
ICAcHBwYFBQQEAwIBAQEBAgMEBAUFBgcHBwgICQj8zAgJCAgHBwcGBQUEBAMCAQON0H/+9BIMCAk
HBAMDAgEBAQEBAgMDBQYHeA4GAwEBAgIDBAUFBwgJCwsNDxABVGwKDxANDxEUCwwMDQ0ODxAQEvz
CCQgICAcHBwYGBAUDAwICAgIDAwUEBgYHBwcICAgJAz4JCAgIBwcHBgYEBQMDAgICAgMDBQQGBgc
HBwgICAAAAgAAAAD7APzAPgBqAAAAR8LFQ8MIy8QKwEPDh8bHQEPFi8WPQE/DTMfEjM/Di8ePQE
/Fh8CBR8HDwMfHjsBPwIfBzM/HTUvBz8CPQEvHiMPAi8HIw8vFhMSDw4LCQgFBAIBAgIDAwg
FBgUGBgcGCAwLCQgHChQLCwsHBwkJCgsNDQwMCwsJCggIBwYFBAMDAQEBAgMEBQcHCAkJCgoLDQ4
TEA8OCwUFAwQCAwEBAgIEBQUHCAgKCgwMDg4PEBESEgsLCwoJCAgHBwUEBAMCAgEBAwQFBwcICQk
KCwsMDQ4PEBESEwsLCgkICAcGBQQDAgIBAwQFBwcICAkKCwsMDQ4PEBESEwsLCgkICAcGBQQDAgI
BAwQFBwcICAkKCwsMDQ4PEBESEwsLCgkICAcGBQQDAgIBAwQFBwcICAkKCwsMDQ4PEBESEwsLCgk
ICAcGBQQDAgIBAwQFBwcICAkKCwsMDQ4PEBESEwsLCgkICAcGBQQDAgIBAwQFBwcICAkKCwsMDQ4
PEBESEwsLCgkICAcGBQQDAgIBAwQFBwcICAkKCwsMDQ4PEBESEwsLCgkICAcGBQQDAgIBAwQFBwc
ICAkKCwsMDQ4PEBESEwsLCgkICAcGBQQDAgIBAwQFBwcICAkKCwsMDQ4PEBESEwsLCgkICAcGBQQ
DAgIB

```
CAgECAgMDBAUFBQYGBgYHCAsLCgkIBwcMBwcHBwoKDAcPERMZDQ0MDAsKCQgHBgUEAwEBAQICAgM
EBAsMDQ8bTSIfGxkMCwsKCQgIBwYFBQMCAgICBAQGBggICQoLDA0NDw8PERERExIUHxwb/bsBAgM
EBQcHCQUDAQEBAQMFBQYICAkLCwwNDg8QEBESEhMUFBUWFRcWGBcYGBYWFRUPDxAQEBEREQ4ODg0
NDQ0MDAwMCwoLCgkJCQgHBwcGBgQFAwMDAgEBAQIDBAUGBgQEAgIDBAUHBwkJCgwMDQ4PDxERERM
TFBQVFRYWFxcYGBgUFRQTEBESEhITFBMODg4NDQ0NDA0LDAsKCwoJCQkIBwcHBgYEBQMDAwIBAzc
ECAoLDAwNDQ4NDg0NBgYGBQYKBQQDAwICAQECBAUHDSEODQoEBAMCAgIBAQICAwMEBQUFBQYGBgY
GCAcHBgYFBQUIBx0GDAgJCgsNDg8JCAkKCgsLCwwPDg0ODQwMDAsLCgoICAgHBgUEBAMCAQEBAgI
EBQYICAYIBwkJCQoKCwsLCgsLCgoHBgYGBQUFBQQEAwMCAQEBAgUGCAkLGg0LCgkICAYDBAMCAQI
DBAQFBgYGBwcIBwkIDQcFBgUEBQgIBgYHEgkJCgoHBgcICAkJCgoLDAwMDg0NDQ0MDAsLCgoKCQg
IBwYGBQQEAwMBAQEBAwRbEhMSEREREBAXFxgYGBgYFxcWFhUVFBQTExEREQ8PDg0MDAoKCAcHBQQ
DAgICAwcGBgUDAwEBAQIDAwMFBAYGBwcHCAkJCQoLCgsMDAwMDQ0NDQ4ODhAQEA8PDw4OGBoZGhg
YFxgWFxUWFRQUExISERAQDw4NDAsLCQkHBgUFAwEBAgIDCggHBgUDAgEBAgMDAwUEBgYGCAcICQk
JCgoLCwwLDQwNDQ0NDg4AAAAACwAAAAD8wOYABEAMwBbAKYAywDTARcBOQFjAZgBoQAAAQ8DMzc
vBisBDwEnDwIdAh8FOwE/BjUvBisBDwEnFwcfBDM/Bic1MxUnNw8GIy8HNyclHwsVIxUfBjsBPwY
1MxcVDw0vCzU/CycVPwMfCR0CDwgjLwQPAREjFSMVIzUjNTcPCxUfDyE/DzUvDiMhIw8BJR8DFQ8
GKwEvBjU/Bx8DFR8KPwUHMzUjFQ8GKwEvBjUjDwcdAR8LOwE/CTUvDg8DFTM1NyMHJyMDIgQDAgJ
CAgECAwQFBQYGCgUG2QQDAgIDBAUFBQYGBQYEBQICAQECAgUEBgUGBgUF6wEBAwUCAwMEBwgEAgE
BAQFFOQEDBAYMDhAQDwgGBgYFAgIEAQECHQoLCgkJCAcFBAMCAXcBAQMDBQQFBhAGBQQEAgIBMwM
BAQMCBAQMBwcICAgIDxAPDg8HBgYEAwMBAQECAgQGBggJCQkJC+UQDg4NDQUGBwcFBAQCAgIDAwU
GBgcICQoLBQwNFAQ50VJFRyAPDQ0LCwkJBwMFAwIBAgQGBwkJCwsNDQ8PDxARAqAQEQ8PDw0NCws
KCAcDBQMCAQIEBgcICgsLDQ0PDxAQEP1gERAPAYoEAgIBAQICBAUGBQYGBgUFBAICAQECAwQFBQU
GBgUGdgEEAgQEBQQGBwgIBwcGBgoKAUw7AQEDAwQEBQUFBQQEAwIBAUC+CggGBgMCAgICBAMECgY
HCQsLCwwMCwoSBwcHCAUEAgEBAgIEAwQHBwgJCgsMDg8NDMpKVlAuLVEBbAMEBB8bBQMEAwICAQE
CCAMDAwOAAwMDAgICAQECAgIDAwOAAwMDAwICAQECF4kgBQUCAQECBAQEAQIDCJrYARsEBAMHBQQ
CAQICAwQEBRoPmw0BAQIDAwQFBgYMDBgvMgYEAwIDAQEBAQMCAwQEGREGBgUFBQQECQQEAwICAQE
BAwYHBQYGBwgJCgpDDwwLCggHBgYDAwMBQFQHBQQBAQICAwUFBQYHBwhwCgkJCAgGBQQDAQEBBAU
LEgEBIib+/yVJBQUGBwcICQkFCgsK9gsLCgoJCQgHBwYFBQMDAQEBAQMDBQUGBwcICQkFCgoL9wo
LCgoJCQgHBwYFBQMDAgID9wQEBAV3BAUDBAMCAQECAwQEBAR3BQQEBAMCAQEBAQJ3GAwRBAUEAwM
DAQEBAQEBAwcKEuCvAwMDAgMBAQEBAwIDAwOvAgUHCAoKDA0OQxgOBwcGCgQEAgMCAQICBQQFBws
KDxZTCgkHBgYGBQYFBQMDAgEBAQIEPayslG9vAAAAABIA3gABAAAAAAAAAAAEAAAABAAAAAAABAAs
AAQABAAAAAAACAAcADAABAAAAAAADAAsAEwABAAAAAAAAEAAsAHgABAAAAAAAFAAsAKQABAAAAAAA
GAAsANAABAAAAAAAKACwAPwABAAAAAAALABIAawADAAEECQAAAAIAfQADAAEECQABABYAfwADAAE
ECQACAA4AlQADAAEECQADABYAowADAAEECQAEABYAuQADAAEECQAFABYAzwADAAEECQAGABYA5QA
DAAEECQAKAFgA+wADAAEECQALACQBUyBTb2NpYWxpY29uc1JlZ3VsYXJTb2NpYWxpY29uc1NvY2l
hbGljb25zVmVyc2lvbiAxLjBTb2NpYWxpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXN
pb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAUwBvAGMAaQBhAGwAaQBjAG8AbgB
zAFIAZQBnAHUAbABhAHIAUwBvAGMAaQBhAGwAaQBjAG8AbgBzAFMAbwBjAGkAYQBsAGkAYwBvAG4
AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAUwBvAGMAaQBhAGwAaQBjAG8AbgBzAEYAbwBuAHQAIAB
nAGUAbgBlAHIAYQB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGU
AdAByAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQA
AAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAsBAgEDAQQBBQEGAQcBCAEJAQoBCwE
MAAh3aGF0c2FwcAd0d2l0dGVyBXZpbWVvCWluc3RhZ3JhbQhsaW5rZWRpbghmYWNlYm9vawtnb29
nbGUtcGx1cwZ0dW1ibHIIc2t5cGUtMDEIeW91dHViZTEAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-tab-icon {
    font-family: 'Socialicons' !important;
}
.e-icons.e-tab-icon {
    position: relative;
    top: 1px;
}
.e-twitter:before {
    content: '\a701';
}
.e-facebook:before {
```

```
    content: '\a705';
}
.e-whatsapp:before {
    content: '\a700';
}
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/icon-position-cs1" %}

See Also

- How to customize selected tab styles

Localization in Vue Tab component

Localization library allows to localize the default text content of Tab. In Tab, The close button's tooltip text alone will be localize based on culture.

| Locale key | en-US (default) |

|------|------|-------------|

| closeButtonTitle | close |

Loading translations

To load translation object in an application use load function of L10n class.

In the below sample, French culture is set to Tab and change the close button's tooltip text.

APP.VUE

```
<template>
  <div id="app">
    <ejs-tab id='element' locale="fr-BE" :showCloseButton=true >
        <e-tabitems>
                    <e-tabitem :header='headerText0'
:content="content0"></e-tabitem>
                    <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                    <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
      </e-tabitems>
    </ejs-tab>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { L10n } from '@syncfusion/ej2-base';
Vue.use(TabPlugin);
L10n.load({
    'fr-BE': {
        'tab': {
        'closeButtonTitle': "Fermer"
        }
    }
});
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
    data: function() {
        return {
          headerText0: { text: 'ASP.NET' },
          headerText1: { text: 'ASP.NET MVC' },
          headerText2: { text: 'JavaScript' },
        content0: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
         content1: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content2: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
        }
   }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/locale-cs1" %}

Orientation in Vue Tab component

This section explains about modifying the position and modes of Tab header.

It allows placing the header section inside the Tab component at different positions by using the headerPlacement property. The available positions are as follows:

- **Top**: Tab header items can be arranged horizontally, and their content can be placed after the header.

Copyright © 2001 -2024 Syncfusion Inc.

- **Bottom**: Tab header items can be arranged horizontally, and their content can be placed before the header.
- **Left**: Tab header items can be arranged vertically, and their content can be placed after the header.
- **Right**: Tab header items can be arranged vertically, and their content can be placed before the header.

It is also adaptable to the available space when the tab items exceed the view space. You can customize the modes by using `overflowMode` property. The available modes are as follows:

- Scrollable
- Popup

APP.VUE

```
<template>
<div id='container'>
        <div id="default" style="padding-top:20px">
            <div class='row' style="float:left">
                    <label> Header Position </label>
                <div>
                    <ejs-dropdownlist id="headerPosition"
ref='headerPosition' :dataSource='headerData' :change='positionChange'
:value='placeValue' :fields='placeFields' ></ejs-dropdownlist>
                </div>
            </div>
            <div class='row' style="float:right">
                    <label> Mode </label>
                <div>
                    <ejs-dropdownlist id="Mode" ref='mode'
:dataSource='modeData' :change='modeChange' :value='value' :fields='fields'
></ejs-dropdownlist>
                </div>
            </div>
        </div>
        <br/>
     <div>
    <ejs-tab id='element' ref="element" heightAdjustMode='None'
height='150px' width='700px'>
        <e-tabitems>
            <e-tabitem :header='headerText0' :content="content0"></e-
tabitem>
            <e-tabitem :header='headerText1' :content="content1"></e-
tabitem>
            <e-tabitem :header='headerText2' :content="content2"></e-
tabitem>
            <e-tabitem :header='headerText3' :content="content3"></e-
tabitem>
            <e-tabitem :header='headerText4' :content="content4"></e-
tabitem>
            <e-tabitem :header='headerText5' :content="content5"></e-
tabitem>
            <e-tabitem :header='headerText6' :content="content6"></e-
tabitem>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <e-tabitem :header='headerText7' :content="content7"></e-
tabitem>
       </e-tabitems>
    </ejs-tab>
        </div>
      </div>
 </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(DropDownListPlugin);
Vue.use(TabPlugin);
export default {
  name: 'app',
     data: function(){
        return {
        modeData: [{ id: 'scrollable', mode: 'Scrollable' }, { id: 'popup',
mode: 'Popup' }],
        headerData: [{ id: 'top', position: 'Top' }, { id: 'bottom',
position: 'Bottom' },{ id: 'left', position: 'Left' }, { id: 'right',
position: 'Right' }],
          fields: { text: 'mode', value: 'id' },
          placeFields: { text: 'position', value: 'id' },
          value: 'scrollable',
          placeValue: 'top',
          headerText0: { text: 'HTML' },
          headerText1: { text: 'C Sharp(C#)' },
          headerText2: { text: 'Java' },
          headerText3: { text: 'VB.Net' },
          headerText4: { text: 'Xamarin' },
          headerText5: { text: 'ASP.NET' },
          headerText6: { text: 'ASP.NET MVC' },
          headerText7: { text: 'JavaScript' },
          content0: 'HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to ' +
        'create web pages. Along with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually ' +
        'engaging web pages, user interfaces for web applications, and user
interfaces for many mobile applications.[1] Web ' +
        'browsers can read HTML files and render them into visible or
audible web pages. HTML describes the structure of a ' +
        'website semantically along with cues for presentation, making it a
markup language, rather than a programming language.',
         content1: 'C# is intended to be a simple, modern, general-purpose,
object-oriented programming language. Its ' +
        'development team is led by Anders Hejlsberg. The most recent
version is C# 5.0, which was released on August 15, 2012.',
        content2: 'Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired ' +
        'by Oracle Corporation, that provides a system for developing
application software and deploying it in a cross-platform ' +
        'computing environment. Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to ' +
        'enterprise servers and supercomputers. While less common, Java
applets run in secure, sandboxed environments to ' +
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        'provide many features of native applications and can be embedded in
HTML pages.',
        content3: 'The command-line compiler, VBC.EXE, is installed as part
of the freeware .NET Framework SDK. Mono also ' +
        'includes a command-line VB.NET compiler. The most recent version is
VB 2012, which was released on August 15, 2012.',
        content4: 'Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers ' +
        'that created Mono,[4] Mono for Android and MonoTouch that are
cross-platform implementations of the Common Language ' +
        'Infrastructure (CLI) and Common Language Specifications (often
called Microsoft .NET). With a C#-shared codebase, ' +
        'developers can use Xamarin tools to write native Android, iOS, and
Windows apps with native user interfaces and share ' +
        'code across multiple platforms.[5] Xamarin has over 1 million
developers in more than 120 countries around the World ' +
        'as of May 2015.',
        content5: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
         content6: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content7: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
        }
        },
        methods:{
        positionChange: function(args) {
           let tabObj=document.getElementById('element');
           let placement =
(document.getElementById('headerPosition')).value;
           tabObj.ej2_instances[0].headerPlacement = placement;
           tabObj.ej2_instances[0].dataBind();
        },
        modeChange: function(args) {
           let tabObj=document.getElementById('element');
           let placement = (document.getElementById('Mode')).value;
           tabObj.ej2_instances[0].overflowMode = placement;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            tabObj.ej2_instances[0].dataBind();
        }
    }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  #app {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }

  .e-content .e-item {
    font-size: 12px;
    margin: 10px;
    text-align: justify;
  }

  #element {
    margin-top: 80px;
    margin-left: 80px;
  }

  #default {
    margin-top: 15px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/orientation-tab-cs3" %}

Drag and drop in Vue Tab component

The Tab component allows you to drag and drop any item by setting allowDragAndDrop to **true**. Items can be reordered to any place by dragging and dropping them onto the desired location.

- If you need to prevent dragging action for a particular item, the onDragStart event can be used which will trigger when the item drag is started. If you need to prevent dropping action for a particular item, the dragged event can be used which will trigger when the drag action is stopped.
- The dragArea defines the area in which the draggable element movement will be occurring. Outside that area will be restricted for the draggable element movement.
- The onDragStart event will be triggered before dragging the item from Tab.
- The dragging event will be triggered when the Tab item is being dragged.
- The dragged event will be triggered when the Tab item is dropped on the target element successfully.

In the following sample, the allowDragAndDrop property is enabled.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="app">
        <div id='container'>
            <ejs-tab id='draggableTab' heightAdjustMode='Auto'
:allowDragAndDrop="true" dragArea="#container">
                <e-tabitems>
                    <e-tabitem :header='headerText0'
:content="content0"></e-tabitem>
                    <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                    <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
                    <e-tabitem :header='headerText3'
:content="content3"></e-tabitem>
                </e-tabitems>
            </ejs-tab>
        </div>
  </div>
</template>
<style>
    #draggableTab .e-content .e-item {
        font-size: 12px;
        margin: 10px;
        text-align: justify;
    }
</style>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(TabPlugin);
export default {
  name: 'app',
   data: function(){
        return {
        headerText0: { text: 'India' },
        headerText1: { text: 'Australia' },
        headerText2: { text: 'USA' },
        headerText3: { text: 'France' },
        content0: 'India officially the Republic of India, is a country in
South Asia. It is the seventh-largest country by area, the second-most
populous country with over 1.2 billion people, and the most populous
democracy in the world. Bounded by the Indian Ocean on the south, the
Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it
shares land borders with Pakistan to the west;China, Nepal, and Bhutan to
the north-east; and Burma and Bangladesh to the east. In the Indian Ocean,
India is in the vicinity of Sri Lanka and the Maldives; in addition, India
Andaman and Nicobar Islands share a maritime border with Thailand and
Indonesia.',
        content1: 'Australia, officially the Commonwealth of Australia, is a
country comprising the mainland of the Australian continent, the island of
Tasmania and numerous smaller islands. It is the world sixth-largest country
by total area. Neighboring countries include Indonesia, East Timor and Papua
New Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to
the north-east; and New Zealand to the south-east.',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        content2: 'The United States of America (USA or U.S.A.), commonly
called the United States (US or U.S.) and America, is a federal republic
consisting of fifty states and a federal district. The 48 contiguous states
and the federal district of Washington, D.C. are in central North America
between Canada and Mexico. The state of Alaska is west of Canada and east of
Russia across the Bering Strait, and the state of Hawaii is in the mid-North
Pacific. The country also has five populated and nine unpopulated
territories in the Pacific and the Caribbean.',
        content3: 'France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.',
        }
    }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/drag-and-drop/default-cs1" %}

Drag and drop item between tabs

It is possible to drag and drop the tab items between two tabs, by manually saving those dropped items as new tab item data through the addTab method of Tab and removing the dragged item through the removeTab method of Tab.

In this example, we have used the tab control as an external source, and the item from the tab component is dragged and dropped onto another Tab. Therefore, it is necessary to use the onDragStart and dragged event of the Tab component, where we can form an event object and save it using the addTab method of the Tab and remove the dragged item through removeTab method of Tab using the dragged item index.

APP.VUE

```
<template>
    <div id="app">
        <div id='container'>
           <ejs-tab ref="firstTabObj" id='firstTab' heightAdjustMode='Auto'
:allowDragAndDrop="true" dragArea="#container"
:onDragStart="firstTabdragStart" :dragged="firstTabDragStop">
                <e-tabitems>
                    <e-tabitem :header='headerText0'
:content="content0"></e-tabitem>
                    <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                    <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
```

```
                    <e-tabitem :header='headerText3'
:content="content3"></e-tabitem>
                </e-tabitems>
            </ejs-tab>
            <ejs-tab ref="secondTabObj" id='secondTab'
heightAdjustMode='Auto' :allowDragAndDrop="true" dragArea="#container"
:onDragStart="secondTabDragStart" :dragged="secondTabDragStop">
                <e-tabitems>
                    <e-tabitem :header='headerText4'
:content="content4"></e-tabitem>
                    <e-tabitem :header='headerText5'
:content="content5"></e-tabitem>
                    <e-tabitem :header='headerText6'
:content="content6"></e-tabitem>
                    <e-tabitem :header='headerText7'
:content="content7"></e-tabitem>
                </e-tabitems>
            </ejs-tab>
        </div>
  </div>
</template>
<style>
    #firstTab .e-content .e-item,
    #secondTab .e-content .e-item {
        font-size: 12px;
        margin: 10px;
        text-align: justify;
    }
    #firstTab {
        margin-bottom: 40px;
    }
    #secondTab {
        margin-top: 40px;
    }
</style>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { isNullOrUndefined } from "@syncfusion/ej2-base";
Vue.use(TabPlugin);
export default {
  name: 'app',
    data: function(){
        return {
        headerText0: { text: 'India' },
        headerText1: { text: 'Australia' },
        headerText2: { text: 'USA' },
        headerText3: { text: 'France' },
        headerText4: { text: 'HTML' },
        headerText5: { text: 'C Sharp(C#)' },
        headerText6: { text: 'Java' },
        headerText7: { text: 'VB.Net' },
        firstTabitem: {},
        secondTabitem: {},
        dragItemContainer: "",
        dragItemIndex: 0,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        content0: 'India officially the Republic of India, is a country in
South Asia. It is the seventh-largest country by area, the second-most
populous country with over 1.2 billion people, and the most populous
democracy in the world. Bounded by the Indian Ocean on the south, the
Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it
shares land borders with Pakistan to the west;China, Nepal, and Bhutan to
the north-east; and Burma and Bangladesh to the east. In the Indian Ocean,
India is in the vicinity of Sri Lanka and the Maldives; in addition, India
Andaman and Nicobar Islands share a maritime border with Thailand and
Indonesia.',
        content1: 'Australia, officially the Commonwealth of Australia, is a
country comprising the mainland of the Australian continent, the island of
Tasmania and numerous smaller islands. It is the world sixth-largest country
by total area. Neighboring countries include Indonesia, East Timor and Papua
New Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to
the north-east; and New Zealand to the south-east.',
        content2: 'The United States of America (USA or U.S.A.), commonly
called the United States (US or U.S.) and America, is a federal republic
consisting of fifty states and a federal district. The 48 contiguous states
and the federal district of Washington, D.C. are in central North America
between Canada and Mexico. The state of Alaska is west of Canada and east of
Russia across the Bering Strait, and the state of Hawaii is in the mid-North
Pacific. The country also has five populated and nine unpopulated
territories in the Pacific and the Caribbean.',
        content3: 'France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.',
        content4: 'HyperText Markup Language, commonly referred to as HTML,
is the standard markup language used to create web pages. Along with CSS,
and JavaScript, HTML is a cornerstone technology, used by most websites to
create visually engaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Web browsers can read
HTML files and render them into visible or audible web pages. HTML describes
the structure of a website semantically along with cues for presentation,
making it a markup language, rather than a programming language.',
        content5: 'C# is intended to be a simple, modern, general-purpose,
object-oriented programming language. Its development team is led by Anders
Hejlsberg. The most recent version is C# 5.0, which was released on August
15, 2012.',
        content6: 'Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired by Oracle Corporation, that
provides a system for developing application software and deploying it in a
cross-platform computing environment. Java is used in a wide variety of
computing platforms from embedded devices and mobile phones to enterprise
servers and supercomputers. While less common, Java applets run in secure,
sandboxed environments to provide many features of native applications and
can be embedded in HTML pages.',
        content7: 'The command-line compiler, VBC.EXE, is installed as part
of the freeware .NET Framework SDK. Mono also includes a command-line VB.NET
compiler. The most recent version is VB 2012, which was released on August
15, 2012.',
        }
    },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    methods: {
        firstTabdragStart: function (args) {
            var firstTabObj = this.$refs.firstTabObj.ej2Instances;
            this.firstTabitem = [firstTabObj.items[args.index]];
            args.draggedItem.style.visibility = 'hidden';
            this.dragItemContainer = args.draggedItem.closest('.e-tab');
        },
        firstTabDragStop: function (args) {
            if (!isNullOrUndefined(args.target.closest('.e-tab')) &&
!this.dragItemContainer.isSameNode(args.target.closest('.e-tab'))) {
                args.cancel = true;
                let TabElement = args.target.closest('.e-tab');
                let dropItem = args.target.closest('.e-toolbar-item');
                if (TabElement != null && dropItem != null) {
                    var firstTabObj = this.$refs.firstTabObj.ej2Instances;
                    var secondTabObj = this.$refs.secondTabObj.ej2Instances;
                    this.dragItemIndex =
Array.prototype.indexOf.call(firstTabObj.element.querySelectorAll('.e-
toolbar-item'), args.draggedItem);
                    let dropItemContainer = args.target.closest('.e-toolbar-
items');
                    let dropItemIndex = (dropItemContainer != null) ?
(Array.prototype.slice.call(dropItemContainer.querySelectorAll('.e-toolbar-
item'))).indexOf(dropItem) : '';
                    secondTabObj.addTab(this.firstTabitem, dropItemIndex);
                    firstTabObj.removeTab(this.dragItemIndex);
                }
            }
        },
        secondTabDragStart: function (args) {
            var secondTabObj = this.$refs.secondTabObj.ej2Instances;
            this.secondTabitem = [secondTabObj.items[args.index]];
            args.draggedItem.style.visibility = 'hidden';
            this.dragItemContainer = args.draggedItem.closest('.e-tab');
        },
        secondTabDragStop: function (args) {
            if (!isNullOrUndefined(args.target.closest('.e-tab')) &&
!this.dragItemContainer.isSameNode(args.target.closest('.e-tab'))) {
                args.cancel = true;
                let TabElement = args.target.closest('.e-tab');
                let dropItem = args.target.closest('.e-toolbar-item');
                if (TabElement != null && dropItem != null) {
                    var firstTabObj = this.$refs.firstTabObj.ej2Instances;
                    var secondTabObj = this.$refs.secondTabObj.ej2Instances;
                    this.dragItemIndex =
Array.prototype.indexOf.call(secondTabObj.element.querySelectorAll('.e-
toolbar-item'), args.draggedItem);
                    let dropItemContainer = args.target.closest('.e-toolbar-
items');
                    let dropItemIndex = (dropItemContainer != null) ?
(Array.prototype.slice.call(dropItemContainer.querySelectorAll('.e-toolbar-
item'))).indexOf(dropItem) : '';
                    firstTabObj.addTab(this.secondTabitem, dropItemIndex);
                    secondTabObj.removeTab(this.dragItemIndex);
                }
            }
        },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/drag-and-drop/tab-to-tab-cs1" %}

Drag and drop items to external source

It is possible to drag and drop the items to any of the external sources from the Tab, by manually saving those dropped items as new node data through the addNodes method of Treeview and removing the dragged item through the removeTab method of Tab.

In this example, we have used the tree view control as an external source, and the item from the tab component is dragged and dropped onto the child nodes of the tree view component. Therefore, it is necessary to use the dragged event of the Tab component, where we can form an event object and save it using the addNodes method of the Treeview and remove the dragged item through the removeTab method of Tab using the dragged item index.

APP.VUE

```
<template>
    <div id="app">
        <div id='container'>
            <ejs-tab ref="tabObj" id='draggableTab' heightAdjustMode='Auto'
:allowDragAndDrop="true" dragArea="#container" :created="onTabCreate"
:dragged="tabDragStop">
                <e-tabitems>
                    <e-tabitem :header='headerText0'
:content="content0"></e-tabitem>
                    <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                    <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
                    <e-tabitem :header='headerText3'
:content="content3"></e-tabitem>
                </e-tabitems>
            </ejs-tab>
            <ejs-treeview ref="treeObj" id="draggableTreeview"
:fields="fields" cssClass="treeview-external-drop-tab">
            </ejs-treeview>
        </div>
  </div>
</template>
<style>
    #draggableTab .e-content .e-item {
        font-size: 12px;
        margin: 10px;
        text-align: justify;
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    .treeview-external-drop-tab .e-list-item,
    .e-bigger .treeview-external-drop-tab .e-list-item {
        border: 0.5px solid #E1E7EC;
        line-height: 15px;
        padding: 0 5px;
    }
    .treeview-external-drop-tab .e-list-item.e-hover>.e-fullrow,
    .treeview-external-drop-tab .e-list-item.e-active>.e-fullrow,
    .treeview-external-drop-tab .e-list-item.e-active.e-hover>.e-fullrow,
    .e-bigger .treeview-external-drop-tab .e-list-item.e-hover>.e-fullrow,
    .e-bigger .treeview-external-drop-tab .e-list-item.e-active>.e-fullrow,
    .e-bigger .treeview-external-drop-tab .e-list-item.e-active.e-hover>.e-
fullrow {
        background-color: transparent;
        border-color: transparent;
        box-shadow: none !important;
    }
    #draggableTab {
        margin-bottom: 40px;
    }
    #draggableTreeview {
        margin-top: 40px;
    }
</style>
<script>
import Vue from 'vue';
import { TabPlugin, TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
import { isNullOrUndefined } from "@syncfusion/ej2-base";
Vue.use(TabPlugin);
Vue.use(TreeViewPlugin);
export default {
  name: 'app',
    data: function(){
        return {
        headerText0: { text: 'India' },
        headerText1: { text: 'Australia' },
        headerText2: { text: 'USA' },
        headerText3: { text: 'France' },
        fields: {
            dataSource: [
                { text: 'Hennessey Venom', id: 'list-01' },
                { text: 'Bugatti Chiron', id: 'list-02' },
                { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
                { text: 'SSC Ultimate Aero', id: 'list-04' },
                { text: 'Koenigsegg CCR', id: 'list-05' },
                { text: 'McLaren F1', id: 'list-06' },
                { text: 'Aston Martin One- 77', id: 'list-07' },
                { text: 'Jaguar XJ220', id: 'list-08' },
                { text: 'McLaren P1', id: 'list-09' },
                { text: 'Ferrari LaFerrari', id: 'list-10' },
            ],
            id: "id", text: "text",
        },
        i: 0,
        content0: 'India officially the Republic of India, is a country in
South Asia. It is the seventh-largest country by area, the second-most
populous country with over 1.2 billion people, and the most populous
```

Copyright © 2001 -2024 Syncfusion Inc.

```
democracy in the world. Bounded by the Indian Ocean on the south, the
Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it
shares land borders with Pakistan to the west;China, Nepal, and Bhutan to
the north-east; and Burma and Bangladesh to the east. In the Indian Ocean,
India is in the vicinity of Sri Lanka and the Maldives; in addition, India
Andaman and Nicobar Islands share a maritime border with Thailand and
Indonesia.',
        content1: 'Australia, officially the Commonwealth of Australia, is a
country comprising the mainland of the Australian continent, the island of
Tasmania and numerous smaller islands. It is the world sixth-largest country
by total area. Neighboring countries include Indonesia, East Timor and Papua
New Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to
the north-east; and New Zealand to the south-east.',
        content2: 'The United States of America (USA or U.S.A.), commonly
called the United States (US or U.S.) and America, is a federal republic
consisting of fifty states and a federal district. The 48 contiguous states
and the federal district of Washington, D.C. are in central North America
between Canada and Mexico. The state of Alaska is west of Canada and east of
Russia across the Bering Strait, and the state of Hawaii is in the mid-North
Pacific. The country also has five populated and nine unpopulated
territories in the Pacific and the Caribbean.',
        content3: 'France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.',
        }
    },
    methods: {
        onTabCreate: function () {
            let tabElement = document.getElementById('draggableTab');
            if (!isNullOrUndefined(tabElement)) {
                tabElement.querySelector('.e-tab-header').classList.add('e-
droppable');
                tabElement.querySelector('.e-content').classList.add('tab-
content');
            }
        },
        tabDragStop: function (args) {
            var tabObj = this.$refs.tabObj.ej2Instances;
            var treeObj = this.$refs.treeObj.ej2Instances;
            let dragTabIndex =
Array.prototype.indexOf.call(tabObj.element.querySelectorAll('.e-toolbar-
item'), args.draggedItem);
            let dragItem = tabObj.items[dragTabIndex];
            let dropNode = args.target.closest('#draggableTreeview .e-list-
item');
            if (dropNode != null && !args.target.closest('#draggableTab .e-
toolbar-item')) {
                args.cancel = true;
                let dropContainer = (document.querySelector('.treeview-
external-drop-tab')).querySelectorAll('.e-list-item');
                let dropIndex = Array.prototype.indexOf.call(dropContainer,
dropNode);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                let newNode = [{ id: 'list' + this.i, text:
dragItem.header.text }];
                tabObj.removeTab(dragTabIndex);
                treeObj.addNodes(newNode, 'Treeview', dropIndex);
            }
        },
    }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/drag-and-drop/tab-to-treeview-cs1" %}

Drag and drop items from external source

It is possible to drag and drop the items from any of the external sources into the Tab, by manually saving those dropped items as new item data through the addTab method of Tab and removing the dragged node through the removeNodes method of Treeview.

In this example, we have used the tree view control as an external source, and the child nodes from the tree view component are dragged and dropped onto the Tab. Therefore, it is necessary to use the nodeDragStop event of the Treeview component, where we can form an event object and save it using the addTab method of the Tab and remove the dragged node through the removeNodes method of Treeview.

APP.VUE

```
<template>
    <div id="app">
        <div id='container'>
            <ejs-tab ref="tabObj" id='draggableTab' heightAdjustMode='Auto'
dragArea="#container">
                <e-tabitems>
                    <e-tabitem :header='headerText0'
:content="content0"></e-tabitem>
                    <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                    <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
                    <e-tabitem :header='headerText3'
:content="content3"></e-tabitem>
                </e-tabitems>
            </ejs-tab>
            <ejs-treeview ref="treeObj" id="draggableTreeview"
:fields="fields" :allowDragAndDrop="true" dragArea="#container"
cssClass="treeview-external-drop-tab" :nodeDragStop="onNodeDragStop"
:nodeDragging="onNodeDrag">
            </ejs-treeview>
        </div>
  </div>
```

```
</template>
<style>
    #draggableTab .e-content .e-item {
        font-size: 12px;
        margin: 10px;
        text-align: justify;
    }
    .treeview-external-drop-tab .e-list-item,
    .e-bigger .treeview-external-drop-tab .e-list-item {
        border: 0.5px solid #E1E7EC;
        line-height: 15px;
        padding: 0 5px;
    }
    .treeview-external-drop-tab .e-list-item.e-hover>.e-fullrow,
    .treeview-external-drop-tab .e-list-item.e-active>.e-fullrow,
    .treeview-external-drop-tab .e-list-item.e-active.e-hover>.e-fullrow,
    .e-bigger .treeview-external-drop-tab .e-list-item.e-hover>.e-fullrow,
    .e-bigger .treeview-external-drop-tab .e-list-item.e-active>.e-fullrow,
    .e-bigger .treeview-external-drop-tab .e-list-item.e-active.e-hover>.e-
fullrow {
        background-color: transparent;
        border-color: transparent;
        box-shadow: none !important;
    }
    #draggableTab {
        margin-bottom: 40px;
    }
    #draggableTreeview {
        margin-top: 40px;
    }
</style>
<script>
import Vue from 'vue';
import { TabPlugin, TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
import { isNullOrUndefined } from "@syncfusion/ej2-base";
Vue.use(TabPlugin);
Vue.use(TreeViewPlugin);
export default {
  name: 'app',
    data: function(){
        return {
        headerText0: { text: 'India' },
        headerText1: { text: 'Australia' },
        headerText2: { text: 'USA' },
        headerText3: { text: 'France' },
        fields: {
            dataSource: [
                { text: 'Hennessey Venom', id: 'list-01' },
                { text: 'Bugatti Chiron', id: 'list-02' },
                { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
                { text: 'SSC Ultimate Aero', id: 'list-04' },
                { text: 'Koenigsegg CCR', id: 'list-05' },
                { text: 'McLaren F1', id: 'list-06' },
                { text: 'Aston Martin One- 77', id: 'list-07' },
                { text: 'Jaguar XJ220', id: 'list-08' },
                { text: 'McLaren P1', id: 'list-09' },
                { text: 'Ferrari LaFerrari', id: 'list-10' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            ],
            id: "id", text: "text",
        },
        content0: 'India officially the Republic of India, is a country in
South Asia. It is the seventh-largest country by area, the second-most
populous country with over 1.2 billion people, and the most populous
democracy in the world. Bounded by the Indian Ocean on the south, the
Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it
shares land borders with Pakistan to the west;China, Nepal, and Bhutan to
the north-east; and Burma and Bangladesh to the east. In the Indian Ocean,
India is in the vicinity of Sri Lanka and the Maldives; in addition, India
Andaman and Nicobar Islands share a maritime border with Thailand and
Indonesia.',
        content1: 'Australia, officially the Commonwealth of Australia, is a
country comprising the mainland of the Australian continent, the island of
Tasmania and numerous smaller islands. It is the world sixth-largest country
by total area. Neighboring countries include Indonesia, East Timor and Papua
New Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to
the north-east; and New Zealand to the south-east.',
        content2: 'The United States of America (USA or U.S.A.), commonly
called the United States (US or U.S.) and America, is a federal republic
consisting of fifty states and a federal district. The 48 contiguous states
and the federal district of Washington, D.C. are in central North America
between Canada and Mexico. The state of Alaska is west of Canada and east of
Russia across the Bering Strait, and the state of Hawaii is in the mid-North
Pacific. The country also has five populated and nine unpopulated
territories in the Pacific and the Caribbean.',
        content3: 'France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.',
        }
    },
    methods: {
        onNodeDragStop: function (args) {
            var tabObj = this.$refs.tabObj.ej2Instances;
            var treeObj = this.$refs.treeObj.ej2Instances;
            let dropElement = args.target.closest('#draggableTab .e-toolbar-
item');
            if (dropElement != null) {
                let tabElement = document.querySelector('#draggableTab');
                let dropItemIndex =
[].slice.call(tabElement.querySelectorAll('.e-toolbar-
item')).indexOf(dropElement);
                let newTabItem = [{
                    header: { 'text': args.draggedNodeData.text.toString()
},
                    content: args.draggedNodeData.text.toString() + '
Content'
                }];
                tabObj.addTab(newTabItem, dropItemIndex);
                treeObj.removeNodes([args.draggedNode]);
                args.cancel = true;
            } else {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                let dropNode = args.target.closest('#draggableTreeview .e-
list-item ');
                if (!isNullOrUndefined(dropNode) && args.dropIndicator ===
'e-drop-in') {
                    args.cancel = true;
                }
            }
        },
        onNodeDrag: function (args) {
            if (!isNullOrUndefined(args.target.closest('.tab-content'))) {
                args.dropIndicator = 'e-no-drop';
            } else if (!isNullOrUndefined(args.target.closest('#draggableTab
.e-tab-header'))) {
                args.dropIndicator = 'e-drop-in';
            }
        },
    }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/drag-and-drop/treeview-to-tab-cs1" %}

Accessibility in Vue Tab component

The Tab component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Tab component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

ARIA attributes

Tab component is designed by considering WAI-ARIA standard. Tab is supported with ARIA Accessibility which is accessible by on-screen readers, and other assistive technology devices.

The following list of attributes are added in the Tab.

| **Roles and Attributes** | **Functionalities** |

| --- | --- |

| tablist | Attribute is set to the Tab header element that describes actual role of the element.|

| tab | Attribute is set to the Tab items element to indicates an interactive element inside a tablist that, when activated, displays its associated tabpanel.|

| tabpanel | Attribute is set to the Tab content that describes the role for viewing the active content.|

| aria-orientation | Attribute is set to the Tab header element indicates the Tab header orientation. Default value of this attribute is horizontal. |

| aria-selected | Attribute set to the Tab items to indicates the selection state for Tab items. Active Tab is set to true for this attribute. |

| aria-labelledby | Attribute is set to the Tab content element to indicates the associated Tab header for the content. |

| aria-controls | Attribute is set to the Tab items element to indicates the associated tabpanel for the header. |

Copyright © 2001 -2024 Syncfusion Inc.

| aria-haspopup | Attribute is set to the Popup element to indicates the popup mode in the Tab. The default value of this attribute is false. If popup mode is enabled, the attribute value is set to true. |

| aria-disabled | Attribute set to the Tab items to It indicates the disabled state of the Tab. |

Keyboard interaction

By default, keyboard navigation is enabled. This component implements keyboard navigation support by following the WAI-ARIA practices. Once focused on the active Tab element, you can use the following key combination for interacting with the Tab.

Key	Description
Left	Moves focus to the previous Tab. If focus is on the first Tab, the focus will not move to any Tab.
Right	Moves focus to the next Tab. If focus is on the last Tab element, the focus will not move to any Tab.
Enter or Space	Selects the Tab if it is not selected. Opens the popup dropdown icon if it is focussed. Select the Tab item as active when popup item is focussed.
Esc(Escape)	Closes the popup if popup is in opened state.
Down or Up	When the popup is open and focused, it will move to previous/next Tab items of the popup in the vertical direction.
Home	Moves focus to the first Tab.
End	Moves focus to the last Tab.
Shift + F10	If popup mode is enabled, it opens the popup when the Tab is focused.
Delete	Deletes the Tab, if close button is enabled in Tab header.
Tab	To Move focus through the interactive elements.
Shift + Tab	To Move focus through the interactive elements.

Ensuring accessibility

The Tab component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Tab component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Tab component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/tab.html" %}

See also

- Accessibility in Syncfusion Vue components

Style in Vue Tab component

The following content provides the exact CSS structure that can be used to modify the control’s appearance based on user preference.

Copyright © 2001 -2024 Syncfusion Inc.

Customizing Tab

Use the following CSS to customize the Tab.

```
.e-tab {
border: 5px solid rgb(173, 255, 47);
}
```

Customizing the Tab items

Use the following CSS to customize the header items of Tab.

```
.e-tab .e-tab-header .e-toolbar-items {
background: #9faed8;
border: 2px solid blue;
}
```

Use the following CSS to customize the content items of Tab.

```
.e-tab .e-content .e-item {
color: #a78515;
font-size: 14px;
}
```

Customizing Tab's header

Use the following CSS to customize the header of Tab control.

```
.e-tab .e-tab-header {
background: #badfba !important;
}
```

Customizing Tab's header icon

Use the following CSS to customize the header item icon of Tab control.

```
.e-tab .e-tab-header .e-toolbar-item .e-tab-icon {
color: #badfba !important;
```

Copyright © 2001 -2024 Syncfusion Inc.

}
`

Customizing Tab's content

Use the following CSS to customize the content of Tab control.

`css
.e-tab .e-content {
background: #d1f6d1 !important;
}
`

Customizing the hover state of Tab control

Use the following CSS to customize the tab item when hovering.

`css
.e-tab .e-tab-header .e-toolbar-item .e-tab-wrap:hover {
background: #d1f6d1 !important;
}
`

Use the following CSS to customize the tab item popup icon when hovering.

`css
.e-tab .e-tab-header .e-hor-nav .e-popup-up-icon:hover,
.e-tab .e-tab-header .e-hor-nav .e-popup-down-icon:hover {
background: #d1f6d1 !important;
}
`

Customizing selected item of Tab control

Use the following CSS to customize the selected tab item.

`css
.e-tab .e-tab-header .e-toolbar-item.e-active {
background: #d1f4d1;
}
`

Use the following CSS to customize the selected tab item text and icon.

`css
.e-tab .e-tab-header .e-toolbar-item.e-active .e-tab-text,
.e-tab .e-tab-header .e-toolbar-item.e-active .e-tab-icon {

Copyright © 2001 -2024 Syncfusion Inc.

color: green !important;

}

`

How To

Load content through post in Vue Tab component

The Tab supports to load external contents through AJAX library. Refer to the following steps.

- Import the Ajax module from ej2-base and initialize with URL path.
- Get the data from Ajax Success event, then initialize the Tab with retrieved external path data.

APP.VUE

```
<template>
   <div id="wrapper" style='margin-top: 20px'>
     <div>
            <ejs-tab id="element" ref=element>
                <e-tabitems>
                    <e-tabitem :header='headerText0'
:content="content0"></e-tabitem>
                    <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                    <e-tabitem :header='headerText2' ></e-tabitem>
                </e-tabitems>
            </ejs-tab>
      </div>
 </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { Ajax } from '@syncfusion/ej2-base';
Vue.use(TabPlugin);
export default {
  name: 'app',
      data: function(){
        return {
        headerText0: { text: 'Facebook' },
        headerText1: { text: 'WhatsApp' },
        headerText2: { text: 'Twitter' },
        content0: 'Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was ' +
        'launched on February 4, 2004, by Mark Zuckerberg with his Harvard
College roommates and fellow students Eduardo ' +
        'Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes.The
founders had initially limited the website\'\s ' +
        'membership to Harvard students, but later expanded it to colleges
in the Boston area, the Ivy League, and Stanford ' +
        'University. It gradually added support for students at various
other universities and later to high-school students.',
         content1: 'WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates ' +
        'under a subscription business model. It uses the Internet to send
text messages, images, video, user location and ' +
```

```
        'audio media messages to other users using standard cellular mobile
numbers. As of February 2016, WhatsApp had a user ' +
        'base of up to one billion,[10] making it the most globally popular
messaging application. WhatsApp Inc., based in ' +
        'Mountain View, California, was acquired by Facebook Inc. on
February 19, 2014, for approximately US$19.3 billion.'
        }
        }, mounted(){
      var ajax: Ajax = new Ajax('./Ajax.html', 'GET', true);
      ajax.send().then();
      ajax.onSuccess = (data: string): void => {
      var obj = this.$refs.element.ej2Instances;
      obj.items[2].content = data;
      obj.refresh();
}
}
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/ajax-cs1" %}

Prevent content swipe selection in Vue Tab component

We can prevent the tab selection on touch swipe action by using the Tab selecting event. Refer the below sample with preventing swipe selection.

APP.VUE

```
<template>
    <div id="app">
   <ejs-tab id="element" :selecting="select">
    <e-tabitems>
                    <e-tabitem :header='headerText0'
:content="content0"></e-tabitem>
                    <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                    <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
                    <e-tabitem :header='headerText3'
:content="content3"></e-tabitem>
                    <e-tabitem :header='headerText4'
:content="content4"></e-tabitem>
                    <e-tabitem :header='headerText5'
:content="content5"></e-tabitem>
                    <e-tabitem :header='headerText6'
:content="content6"></e-tabitem>
                    <e-tabitem :header='headerText7'
:content="content7"></e-tabitem>
            </e-tabitems>
        </ejs-tab>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(TabPlugin);
export default {
  name: 'app',
     data: function(){
        return {
          headerText0: { text: 'HTML' },
          headerText1: { text: 'C Sharp(C#)' },
          headerText2: { text: 'Java' },
          headerText3: { text: 'VB.Net' },
          headerText4: { text: 'Xamarin' },
          headerText5: { text: 'ASP.NET' },
          headerText6: { text: 'ASP.NET MVC' },
          headerText7: { text: 'JavaScript' },
          content0: 'HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to ' +
        'create web pages. Along with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually ' +
        'engaging web pages, user interfaces for web applications, and user
interfaces for many mobile applications.[1] Web ' +
        'browsers can read HTML files and render them into visible or
audible web pages. HTML describes the structure of a ' +
        'website semantically along with cues for presentation, making it a
markup language, rather than a programming language.',
         content1: 'C# is intended to be a simple, modern, general-purpose,
object-oriented programming language. Its ' +
        'development team is led by Anders Hejlsberg. The most recent
version is C# 5.0, which was released on August 15, 2012.',
        content2: 'Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired ' +
        'by Oracle Corporation, that provides a system for developing
application software and deploying it in a cross-platform ' +
        'computing environment. Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to ' +
        'enterprise servers and supercomputers. While less common, Java
applets run in secure, sandboxed environments to ' +
        'provide many features of native applications and can be embedded in
HTML pages.',
        content3: 'The command-line compiler, VBC.EXE, is installed as part
of the freeware .NET Framework SDK. Mono also ' +
        'includes a command-line VB.NET compiler. The most recent version is
VB 2012, which was released on August 15, 2012.',
        content4: 'Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers ' +
        'that created Mono,[4] Mono for Android and MonoTouch that are
cross-platform implementations of the Common Language ' +
        'Infrastructure (CLI) and Common Language Specifications (often
called Microsoft .NET). With a C#-shared codebase, ' +
        'developers can use Xamarin tools to write native Android, iOS, and
Windows apps with native user interfaces and share ' +
        'code across multiple platforms.[5] Xamarin has over 1 million
developers in more than 120 countries around the World ' +
        'as of May 2015.',
```

```
        content5: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
         content6: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content7: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
        } }, methods:{
     select: function(e) {
      if (e.isSwiped) {
        e.cancel = true;
      }
      }
    }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/swipe-selection-cs1" %}

Customize selected tab styles in Vue Tab component

You can customize the Tab style by overriding its header and active tab CSS classes. Define HTML string for adding animation and customizing

the Tab header and pass it to text property. Now you can override the style using custom CSS classes added to

the Tab elements.

APP.VUE

```
<template>
```

```
    <div id="wrapper" style='margin-top: 20px'>
        <div>
            <ejs-tab id="element" :items='items'></ejs-tab>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(TabPlugin);
export default {
    name: 'app',
    data: function () {
        return {
            items: [
                {
                    header: { 'text': '<div id="template-wrap"><div
class="e-image e-andrew"></div><div class="e-title fade-
in">Andrew</div></div>' },
                    content: 'Andrew received his BTS commercial in 1974 and
a Ph.D. in international marketing from the University of Dallas in 1981.He
is fluent in French and Italian and reads German.He joined the company as a
sales representative, was promoted to sales manager in January 1992 and to
vice president of sales in March 1993.Andrew is a member of the Sales
Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim
Importers Association.'
                },
                {
                    header: { 'text': '<div id="template-wrap"><div
class="e-image e-margaret"></div><div class="e-title fade-
in">Margaret</div></div>' },
                    content: 'Margaret holds a BA in English literature from
Concordia College (1958) and an MA from the American Institute of Culinary
Arts (1966).She was assigned to the London office temporarily from July
through November 1992.'
                },
                {
                    header: { 'text': '<div id="template-wrap"><div
class="e-image e-janet"></div><div class="e-title fade-
in">Janet</div></div>' },
                    content: 'Janet has a BS degree in chemistry from Boston
College (1984).She has also completed a certificate program in food
retailing management.Janet was hired as a sales associate in 1991 and
promoted to sales representative in February 1992.'
                }
            ]
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
#template-wrap {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    display: flex;
}
.e-content .e-item {
    font-size: 12px;
    margin: 10px;
    text-align: justify;
}
.container {
    min-width: 350px;
    max-width: 500px;
    margin: 0 auto;
}
.e-image {
    background-size: 33px;
    width: 33px;
    height: 33px;
    margin: 0 auto;
}
.e-image.e-andrew { /* csslint allow: adjoining-classes */
    background-image:
url('https://ej2.syncfusion.com/demos/src/images/employees/3.png');
}
.e-image.e-margaret { /* csslint allow: adjoining-classes */
    background-image:
url('https://ej2.syncfusion.com/demos/src/images/employees/6.png');
}
.e-image.e-janet { /* csslint allow: adjoining-classes */
    background-image:
url('https://ej2.syncfusion.com/demos/src/images/employees/7.png');
}
.e-tab .e-toolbar-item .e-title {
    margin-top: 8px;
}
.e-toolbar .e-toolbar-items .e-toolbar-item:not(.e-separator),
.e-toolbar .e-toolbar-items .e-toolbar-item .e-tab-wrap {
    width: 125px;
    height: 50px;
}
.e-tab .e-tab-header .e-toolbar-item.e-active .e-tab-wrap { /* csslint
allow: adjoining-classes */
    background-color: #08c;
}
.e-tab .e-tab-header {
    background-color: #e6e6e6;
}
.e-tab .e-tab-header .e-toolbar-item:not(.e-active) .e-tab-wrap:hover {
    background-color: #f2f2f2;
    color: #000;
}
.e-tab .e-toolbar .e-toolbar-items .e-toolbar-item .e-text-wrap {
    height: 50px;
}
.e-tab .e-toolbar-item.e-active .e-title { /* csslint allow: adjoining-
classes */
    display: block;
    color: white;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.e-tab .e-toolbar-item .e-text-wrap,
.e-tab .e-toolbar-item .e-tab-text {
    width: inherit;
    text-align: center;
}
.e-tab .e-toolbar-item.e-active .e-title.fade-in {
    opacity: 1;
    animation-name: fadeInOpacity;
    animation-iteration-count: 1;
    animation-timing-function: ease-in;
    animation-duration: 0.5s;
}
@keyframes fadeInOpacity {
    0% {
        opacity: 0;
    }
    100% {
        opacity: 1;
    }
}
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/customize-tab-cs1" %}

Customize tab scroll step in Vue Tab component

Tab supports to customize the scrolling distance when you click the left and right side navigation icons. we can customize ScrollStep property for scrolling distance. Refer to the following code example.

By using Tab scrollStep property, pass a required value to customize tab scrollStep.

APP.VUE

```
<template>
    <div id="app">
    <ejs-tab id='element' scrollStep='50'>
        <e-tabitems>
          <e-tabitem :header='headerText0' :content="content0"></e-tabitem>
          <e-tabitem :header='headerText1' :content="content1"></e-tabitem>
          <e-tabitem :header='headerText2' :content="content2"></e-tabitem>
          <e-tabitem :header='headerText3' :content="content3"></e-tabitem>
          <e-tabitem :header='headerText4' :content="content4"></e-tabitem>
          <e-tabitem :header='headerText5' :content="content5"></e-tabitem>
          <e-tabitem :header='headerText6' :content="content6"></e-tabitem>
          <e-tabitem :header='headerText7' :content="content7"></e-tabitem>
      </e-tabitems>
    </ejs-tab>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(TabPlugin);
export default {
  name: 'app',
   data: function(){
        return {
```

```
            headerText0: { text: 'HTML' },
            headerText1: { text: 'C Sharp(C#)' },
            headerText2: { text: 'Java' },
            headerText3: { text: 'VB.Net' },
            headerText4: { text: 'Xamarin' },
            headerText5: { text: 'ASP.NET' },
            headerText6: { text: 'ASP.NET MVC' },
            headerText7: { text: 'JavaScript' },
          content0: 'HyperText Markup Language, commonly referred to as HTML,
is the standard markup language used to ' +
          'create web pages. Along with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually ' +
          'engaging web pages, user interfaces for web applications, and user
interfaces for many mobile applications.[1] Web ' +
          'browsers can read HTML files and render them into visible or
audible web pages. HTML describes the structure of a ' +
          'website semantically along with cues for presentation, making it a
markup language, rather than a programming language.',
          content1: 'C# is intended to be a simple, modern, general-purpose,
object-oriented programming language. Its ' +
          'development team is led by Anders Hejlsberg. The most recent
version is C# 5.0, which was released on August 15, 2012.',
          content2: 'Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired ' +
          'by Oracle Corporation, that provides a system for developing
application software and deploying it in a cross-platform ' +
          'computing environment. Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to ' +
          'enterprise servers and supercomputers. While less common, Java
applets run in secure, sandboxed environments to ' +
          'provide many features of native applications and can be embedded in
HTML pages.',
          content3: 'The command-line compiler, VBC.EXE, is installed as part
of the freeware .NET Framework SDK. Mono also ' +
          'includes a command-line VB.NET compiler. The most recent version is
VB 2012, which was released on August 15, 2012.',
          content4: 'Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers ' +
          'that created Mono,[4] Mono for Android and MonoTouch that are
cross-platform implementations of the Common Language ' +
          'Infrastructure (CLI) and Common Language Specifications (often
called Microsoft .NET). With a C#-shared codebase, ' +
          'developers can use Xamarin tools to write native Android, iOS, and
Windows apps with native user interfaces and share ' +
          'code across multiple platforms.[5] Xamarin has over 1 million
developers in more than 120 countries around the World ' +
          'as of May 2015.',
          content5: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
          'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
          'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
          'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
          'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        'ASP.NET components to process SOAP messages.',
         content6: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content7: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
        }
   }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/scrollstep-cs1" %}

Create wizard using tab in Vue Tab component

Tab items can be disabled dynamically by passing the index and boolean value with the enableTab method.

In the below Wizard sample, each Tab is integrated with required components to complete the reservation. Each field is provided with validation

for all mandatory option to proceed to next tabs. Using Tab item's template property the components are added intocontent.

Create the following contents for each tab in the wizard.

1. **Search tab:**

Created with DropDownList to select the source, destination and type of ticket. A DatePicker for choosing the date of journey.

2. **Train tab:**

Based on the selected start and end point, populated Grid with random list of available seats and train list. Initially define the columns

and row selected event for validating, after the source and destination chosen update the dataSource

Copyright © 2001 -2024 Syncfusion Inc.

for the Grid.

3. **Passenger tab:**

A table with Textbox, Numeric, DropDownList for adding passenger name, age, gender and preferred berth/seat. Add validation on entering

passenger details to proceed.

4. **Payment tab:**

Calculate the ticket cost based on location, passenger count and ticket type. Generate data for Grid with passenger details, train number

and ticket cost summary.

You can go back on each tab using buttons available in it and tabs are disabled to navigate through tab header click actions. Once you end

the wizard all the data get cleared and wizard goes back to starting tab.

APP.VUE

```
<template>
  <div class="col-lg-12 control-section e-tab-section">
    <div class="e-sample-resize-container" id="dialog_target">
        <div id='booking' style='display: none'>
            <div class="wizard-title">Plan your journey</div>
            <div class="responsive-align">
                <div class='row'>
                    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-
item">
                        <ejs-dropdownlist id="startPoint" width="100%"
:dataSource='cities' :fields='autoCompleteFields' floatLabelType='Auto'
placeholder='From'></ejs-dropdownlist>
                    </div>
                    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-
item">
                        <ejs-dropdownlist id="endPoint" width="100%"
:dataSource='cities' :fields='autoCompleteFields' floatLabelType='Auto'
placeholder='To'></ejs-dropdownlist>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-
item">
                        <ejs-datepicker id="date" width='100%'
placeholder='Journey Date' floatLabelType='Auto' :min='dateMin'
:max='dateMax' :focus='showDate'></ejs-datepicker>
                    </div>
                    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-
item">
                        <ejs-dropdownlist id="quota" :dataSource='quotas'
placeholder='Ticket type' floatLabelType='Auto' :fields='fields'></ejs-
dropdownlist>
                    </div>
```

```
                </div>
                <div class="btn-container">
                    <button id='searchNext' class='e-btn' v-
on:click='btnClicked'>Search Train</button>
                </div>
                <span id="err1"></span>
            </div>
        </div>
        <div id='selectTrain' style='display: none'>
            <div class="wizard-title">Select the train from the list </div>
            <ejs-grid id='availableTrain' width="100%"
:rowSelected='trainSelected'>
                <e-columns>
                    <e-column field='TrainNo' headerText='Train No'
width=120 type='number'></e-column>
                    <e-column field='Name' headerText='Name' width=140></e-
column>
                    <e-column field='Departure' headerText='Departure'
width=120></e-column>
                    <e-column field='Arrival' headerText='Arrival'
width=140></e-column>
                    <e-column field='Availability' headerText='Availability'
width=140 type='number'></e-column>
                </e-columns>
            </ejs-grid>
            <br />
            <div class="btn-container">
                <button id="goToSearch" class='e-btn' v-
on:click='btnClicked'>Back</button>
                <button id="bookTickets" class='e-btn' v-
on:click='btnClicked'>Continue</button>
            </div>
            <span id="err2"></span>
        </div>
        <div id='details' style='display: none'>
            <div class="details-page wizard-title">Enter the passenger
details</div>
            <div id="PassengersList">
                <table id="passenger-table">
                    <colgroup>
                        <col>
                        <col>
                        <col>
                        <col>
                        <col>
                        <col>
                    </colgroup>
                    <thead>
                        <tr>
                            <th class="name-header">Name</th>
                            <th class="age-header">Age</th>
                            <th class="gender-header">Gender</th>
                            <th class="type-header">Berth Preference</th>
                        </tr>
                    </thead>
                    <tbody>
                        <tr>
```

```
                            <td>
                                <input id='pass_name1' class="e-input"
type="text" placeholder="Passenger Name">
                            </td>
                            <td>
                                <ejs-numerictextbox id='pass_age1'
showSpinButton=false min=1 max=100 value=18 format=n0></ejs-numerictextbox>
                            </td>
                            <td>
                                <ejs-dropdownlist id='pass_gender1'
:dataSource='gender' text="Male" :fields='fields'></ejs-dropdownlist>
                            </td>
                            <td>
                                <ejs-dropdownlist id='pass_berth1'
:dataSource='berths' placeholder="Optional" :fields='fields'></ejs-
dropdownlist>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <input id='pass_name2' class="e-input"
type="text" placeholder="Passenger Name">
                            </td>
                            <td>
                                <ejs-numerictextbox showSpinButton=false
min=1 max=100 value=18 format=n0></ejs-numerictextbox>
                            </td>
                            <td>
                                <ejs-dropdownlist id='pass_gender2'
:dataSource='gender' text="Male" :fields='fields'></ejs-dropdownlist>
                            </td>
                            <td>
                                <ejs-dropdownlist id='pass_berth2'
:dataSource='berths' placeholder="Optional" :fields='fields'></ejs-
dropdownlist>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <input id='pass_name3' class="e-input"
type="text" placeholder="Passenger Name">
                            </td>
                            <td>
                                <ejs-numerictextbox showSpinButton=false
min=1 max=100 value=18 format=n0></ejs-numerictextbox>
                            </td>
                            <td>
                                <ejs-dropdownlist id='pass_gender3'
:dataSource='gender' text="Male" :fields='fields'></ejs-dropdownlist>
                            </td>
                            <td>
                                <ejs-dropdownlist id='pass_berth3'
:dataSource='berths' placeholder="Optional" :fields='fields'></ejs-
dropdownlist>
                            </td>
                        </tr>
                    </tbody>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                </table>
            </div>
            <br />
            <div class="btn-container">
                <button id="goBackToBook" class='e-btn' v-
on:click='btnClicked'>Back</button>
                <button id="confirmTickets" class='e-btn' v-
on:click='btnClicked'>Continue</button>
            </div>
            <span id="err3"></span>
        </div>
        <div id='confirm' style='display: none'>
            <div class="tab-title1 wizard-title">Confirm the details and
proceed</div>
            <ejs-grid id='ticketDetailGrid' width="100%">
                <e-columns>
                    <e-column field='TrainNo' headerText='Train No'
width=120 type='number'></e-column>
                    <e-column field='PassName' headerText='Name'
width=140></e-column>
                    <e-column field='Gender' headerText='Gender'
width=120></e-column>
                    <e-column field='Berth' headerText='Berth'
width=140></e-column>
                </e-columns>
            </ejs-grid>
            <br />
            <div id="amount"></div>
            <br />
            <div class="btn-container">
                <button id="goBackDetails" class='e-btn' v-
on:click='btnClicked'>Back</button>
                <button id="makePayment" class='e-btn' v-
on:click='btnClicked'>Pay</button>
            </div>
        </div>
        <ejs-tab id="tab_wizard" heightAdjustMode="None" height=390
showCloseButton=false :select="tabSelected">
            <e-tabitems>
                <e-tabitem :header='headerText0' content="#booking"></e-
tabitem>
                <e-tabitem :header='headerText1' content="#selectTrain"
disabled=true></e-tabitem>
                <e-tabitem :header='headerText2' content="#details"
disabled=true></e-tabitem>
                <e-tabitem :header='headerText3' content="#confirm"
disabled=true></e-tabitem>
            </e-tabitems>
        </ejs-tab>
        <div>
            <ejs-dialog id='alertDialog' header='Success' width=250
isModal=true showCloseIcon=true content='Your payment successfully
processed' target='#dialog_target' :created='dlgCreated'
visible=false></ejs-dialog>
        </div>
    </div>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { isNullOrUndefined } from '@syncfusion/ej2-base';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { GridPlugin } from '@syncfusion/ej2-vue-grids';
import { NumericTextBoxPlugin} from '@syncfusion/ej2-vue-inputs';
import { DialogPlugin} from '@syncfusion/ej2-vue-popups';
import { DatePickerPlugin } from '@syncfusion/ej2-vue-calendars';
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(NumericTextBoxPlugin);
Vue.use(DropDownListPlugin);
Vue.use(DatePickerPlugin);
Vue.use(GridPlugin);
Vue.use(TabPlugin);
Vue.use(DialogPlugin);
export default {
  name: 'app',
     data: function() {
    return {
        cities: [
            { name: 'Chicago', fare: 300 },
            { name: 'San Francisco', fare: 125 },
            { name: 'Los Angeles', fare: 175 },
            { name: 'Seattle', fare: 250 },
            { name: 'Florida', fare: 150 }
        ],
        autoCompleteFields: { text: 'name', value: 'name' },
        dateMin: new Date(new Date().getTime()),
        dateMax: new Date(new Date(new Date().getTime() + 60 * 24 * 60 * 60
* 1000)),
        quotas: [
            { id: '1', text: 'Business Class' },
            { id: '2', text: 'Economy Class' },
            { id: '4', text: 'Common Class' }
        ],
        fields: { id: 'id', text: 'text', value: 'text' },
        gender: [
            { id: '1', text: 'Male' },
            { id: '2', text: 'Female' }
        ],
        berths: [
            { id: '1', text: 'Upper' },
            { id: '2', text: 'Lower' },
            { id: '3', text: 'Middle' },
            { id: '4', text: 'Window' },
            { id: '5', text: 'Aisle' }
        ],
       headerText0: { 'text': 'New Booking' },
       headerText1: { 'text': 'Train List' },
       headerText2: { 'text': 'Add Passenger' },
       headerText3: { 'text': 'Make Payment' }
    }
  },
   methods: {
       showDate: function(args){
```

```
            var journeyDate =
document.getElementById("date").ej2_instances[0];
            journeyDate.show();
        },
        btnClicked: function(args){
           var startPoint = document.getElementById("startPoint");
           var endPoint = document.getElementById("endPoint");
           let tabObj=document.getElementById('tab_wizard');
             switch (args.target.id) {
                  case 'searchNext':
                       if (!isNullOrUndefined(startPoint.value) &&
!isNullOrUndefined(endPoint.value) &&
!isNullOrUndefined(document.getElementById("quota").value) &&
!isNullOrUndefined(document.getElementById("date"))) {
                       if (!isNullOrUndefined(startPoint.value) &&
startPoint.value === endPoint.value) {
                            document.getElementById('err1').innerText = '*
Arrival point can\'t be same as Departure';
                        } else {
                             tabObj.ej2_instances[0].enableTab(0, false);
                             tabObj.ej2_instances[0].enableTab(1, true);
                             this.filterTrains(args);
                             tabObj.ej2_instances[0].select(1);
                             document.getElementById('err1').innerText = '';
                             document.getElementById('err2').innerText = '';
                        }
                      } else {
                           document.getElementById('err1').innerText = '*
Please fill all the details before proceeding';
                      }
                     break;
                case 'bookTickets':
                     if
(document.getElementById('availableTrain').ej2_instances[0].getSelectedRecor
ds() === undefined ||
document.getElementById('availableTrain').ej2_instances[0].getSelectedRecord
s().length === 0) {
                           document.getElementById('err2').innerText = '*
Select your convenient train';
                       } else {
                             tabObj.ej2_instances[0].enableTab(2, true);
                             tabObj.ej2_instances[0].select(2);
                             tabObj.ej2_instances[0].enableTab(1, false);
                             document.getElementById('err2').innerText = '';
                        }
                       break;
                case 'confirmTickets':
                     let name = document.getElementById('pass_name1');
                     let age = document.getElementById('pass_age1');
                     let gender = document.getElementById('pass_gender1');
                     if (name.value === '' || age.value === '' ||
gender.value === '') {
                           document.getElementById('err3').innerText = '*
Please enter passenger details';
                       } else {
                             tabObj.ej2_instances[0].enableTab(3, true);
                             tabObj.ej2_instances[0].select(3);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                            tabObj.ej2_instances[0].enableTab(2, false);
                            document.getElementById('err3').innerText = '';
                            this.finalizeDetails(args);
                        }
                        break;
                    case 'makePayment':

document.getElementById('alertDialog').ej2_instances[0].show();
                        break;
                    case 'goToSearch':
                            this.selectedTrain = [];
                            tabObj.ej2_instances[0].enableTab(0, true);
                            tabObj.ej2_instances[0].select(0);
                            tabObj.ej2_instances[0].enableTab(1, false);
                            break;
                    case 'goBackToBook':
                            tabObj.ej2_instances[0].enableTab(1, true);
                            tabObj.ej2_instances[0].select(1);
                            tabObj.ej2_instances[0].enableTab(2, false);
                            break;
                    case 'goBackDetails':
                            tabObj.ej2_instances[0].enableTab(2, true);
                            tabObj.ej2_instances[0].select(2);
                            tabObj.ej2_instances[0].enableTab(3, false);
                            break;
                    }
             },
            trainSelected: function(args){
                    this.selectedTrain = args.data;
             },
            tabSelected: function(args){
                    if (args.isSwiped)
                         args.cancel = true;
            },
            dlgCreated: function(args){
                    let tabObj=document.getElementById('tab_wizard');
                    let alertDlg =
document.getElementById("alertDialog").ej2_instances[0];
                    alertDlg.hide();
                    alertDlg.content = "Your payment successflly processed";
                    alertDlg.buttons = [{
                    buttonModel: { content: "Ok", isPrimary: true },
                      click: function () {

document.getElementById("alertDialog").ej2_instances[0].hide();
                            tabObj.ej2_instances[0].enableTab(0, true);
                            tabObj.ej2_instances[0].enableTab(1, false);
                            tabObj.ej2_instances[0].enableTab(2, false);
                            tabObj.ej2_instances[0].enableTab(3, false);
                            tabObj.ej2_instances[0].select(0);
                      }
             }];
             alertDlg.dataBind();
             alertDlg.hide();
             },
             filterTrains: function(args){
                 let result= [];
```

```
                let fromCity = document.getElementById('startPoint').text;
                let toCity=document.getElementById('endPoint').text;
                let count= Math.floor((Math.random() * 3) + 2);
                for (var i = 0; i < count; i++) {
                    let details= [];
                    details.TrainNo = Math.floor((Math.random() * 20) +
19000);
                    details.Name = 'Train ' + i;
                    details.Departure = fromCity;
                    details.Arrival = toCity;
                    details.Availability = Math.floor((Math.random() * 20) +
20);
                    result.push(details);
                }

document.getElementById('availableTrain').ej2_instances[0].dataSource =
result;
          },
          finalizeDetails: function(args){
               var reserved= [];
               var passCount = 0;
               var startPoint = document.getElementById("startPoint");
               var endPoint = document.getElementById("endPoint");
               for (let i = 1; i <= 3; i++) {
                    let name = document.getElementById('pass_name' + i);
                    let berthSelected= document.getElementById('pass_berth'
+ i);
                    let gender = document.getElementById('pass_gender' + i);
                    if (name.value !== '') {
                        let details = [];
                        let berth = berthSelected.value;
                        details.TrainNo =
this.selectedTrain.TrainNo.toString();
                        details.PassName = name.value;
                        details.Gender = gender.value;
                        details.Berth = (berth === '') ? 'Any' : berth;
                        reserved.push(details);
                        passCount++;
                   }
                   let calcFare= 0;
               for (let i= 0; i < this.cities; i++) {
                   if (startPoint.value === this.cities[i].name) {
                       calcFare = calcFare + this.cities[i].fare;
                    }
                    if (endPoint.value === this.cities[i].name) {
                        calcFare = calcFare + this.cities[i].fare;
                    }
                }
                let displayAmt = document.getElementById('amount');
                if (document.getElementById('quota').value === 'Economy
Class') {
                     displayAmt.innerText = "Total payable amount: $" +
passCount * (300 + calcFare);
                } else if (document.getElementById("quota").value ===
'Business Class') {
                     displayAmt.innerText = "Total payable amount: $" +
passCount * (500 + calcFare);
```

```
                } else if (document.getElementById("quota").value ===
'Common Class') {
                    displayAmt.innerText = "Total payable amount: $" +
passCount * (150 + calcFare);
                }
            }

document.getElementById('ticketDetailGrid').ej2_instances[0].dataSource =
reserved;
        }
    }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
    .e-content .e-item {
        font-size: 12px;
        margin: 10px;
    }
    #amount {
        text-align: right;
        font-size: 15px;
        padding: 15px 0px;
    }
    #passenger-table th {
        text-align: center;
        font-size: 14px;
        font-weight: 400;
        border: 1px solid gainsboro;
    }
    #passenger-table td,
    th {
        padding: 10px;
    }
    #passenger-table td {
        border: 1px solid gainsboro;
    }
    .name-header {
        width: 200px;
    }
    .age-header {
        width: 80px;
    }
    .gender-header {
        width: 120px;
    }
    .type-header {
```

```
        width: 150px;
    }
    .btn-container {
        text-align: center;
    }
    .search-item {
        padding-right: 50px;
        padding-bottom: 20px;
    }
    .item-title {
        font-weight: 500;
        padding-top: 10px;
    }
    @media (max-width: 450px) {
        .e-sample-resize-container {
            height: 450px;
        }
        .responsive-align {
            width: 75%;
            margin: 0 auto;
        }
        .search-item {
            padding: 0 0 20px 0;
            width: 100%;
        }
    }
    #err1,
    #err2,
    #err3 {
        font-weight: bold;
        color: red;
        display: block;
        margin-top: 15px;
    }
    .wizard-title {
        font-size: 15px;
    }
    #PassengersList {
        overflow: auto;
    }
    #passenger-table {
        min-width: 500px;
        width: 100%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/wizard-cs1" %}

Load tab with data source in Vue Tab component

You can bind any data object to Tab items, by mapping it to a header and content property.

In the below demo, Data is fetched from an `OData` service using `DataManager`. The result data is formatted as a JSON object with `header`

and `content` fields, which is set to items property of Tab.

APP.VUE

```
<template>
    <div id="app">
    <ejs-tab ref="TabInstance">
    </ejs-tab>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { DataManager, Query, ODataV4Adaptor, ReturnOption } from
'@syncfusion/ej2-data';
Vue.use(TabPlugin);
export default {
  name: 'app',
  data () {
}, mounted(){
  new DataManager({ url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Employees', adaptor:
new ODataV4Adaptor}).executeQuery(new Query().range(4, 7)).then((e) => {
        var result = e.result;
        var obj = this.$refs.TabInstance.ej2Instances
        var itemsData =[];
        var  mapping =  { header: 'FirstName', content: 'Notes' };
        for(var i= 0; i < result.length; i++) {
        itemsData.push({ header: {text: result[i][mapping.header]}, content:
result[i][mapping.content] });
       }
      obj.items = itemsData;
      obj.refresh();
});
}
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/datasource-cs1" %}

Add nested tabs in Vue Tab component

Tab supports to render the nested level of Tabs by using `content` property. You can add the nested Tab element inside the parent Tab `content`

property. To render the nested Tab, initialize the component using the id of Tab from a selected event handler.

APP.VUE

```
<template>
    <div id="app">
```

```
    <ejs-tab ref='element'  :selected= "handleSelectEvent" :created=
"handleCreatedEvent">
    <e-tabitems>
              <e-tabitem :header='headerText0' :content="content0"></e-
tabitem>
              <e-tabitem :header='headerText1' :content="content1"></e-
tabitem>
              <e-tabitem :header='headerText2' :content="content2"></e-
tabitem>
    <e-tabitems>
    </ejs-tab>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { Tab, TabComponent, SelectEventArgs } from '@syncfusion/ej2-
navigations';
import { enableRipple, isNullOrUndefined as isNOU } from '@syncfusion/ej2-
base';
Vue.use(TabPlugin);
export default {
  name: 'app',
   data: function(){
        return {
          headerText0: { text: 'USA' },
          headerText1: { text: 'France' },
          headerText2: { text: 'Australia' },
         content0: '<div id="usa_tab"></div>',
         content1: '<div id="france_tab"></div>',
         content2: '<div id="australia_tab"></div>',
}
  },methods:{
handleCreatedEvent: function() {
  if (isNOU(document.querySelector('#usa_tab.e-tab'))) {
    var usa_obj = new Tab({
       items:[
  {
    header: { 'text': 'New York' },
    content: 'New York City comprises 5 boroughs sitting where the Hudson
River meets the Atlantic Ocean. At its core is Manhattan, a densely
populated borough that's among the world's major commercial, financial and
cultural centers. Its iconic sites include skyscrapers such as the Empire
State Building and sprawling Central Park. Broadway theater is staged in
neon-lit Times Square.'
  },
  {
    header: { 'text': 'Los Angeles' },
    content: 'Los Angeles is a sprawling Southern California city and the
center of the nation's film and television industry. Near its iconic
Hollywood sign, studios such as Paramount Pictures, Universal and Warner
Brothers offer behind-the-scenes tours. On Hollywood Boulevard, TCL Chinese
Theatre displays celebrities' hand- and footprints, the Walk of Fame honors
thousands of luminaries and vendors sell maps to stars' homes.'
  },
  {
    header: { 'text': 'Chicago' },
```

```
    content: 'Chicago, on Lake Michigan in Illinois, is among the largest
cities in the U.S. Famed for its bold architecture, it has a skyline
punctuated by skyscrapers such as the iconic John Hancock Center, 1,451-ft.
Willis Tower (formerly the Sears Tower) and the neo-Gothic Tribune Tower.
The city is also renowned for its museums, including the Art Institute of
Chicago with its noted Impressionist and Post-Impressionist works.'
  }
]
    }, '#usa_tab');
  }
}
handleSelectEvent: function(e) {
  if (e.selectedIndex === 0 && isNOU(document.querySelector('#usa_tab.e-
tab'))) {
    var usa_obj = new Tab({
       items:[
  {
    header: { 'text': 'New York' },
    content: 'New York City comprises 5 boroughs sitting where the Hudson
River meets the Atlantic Ocean. At its core is Manhattan, a densely
populated borough that's among the world's major commercial, financial and
cultural centers. Its iconic sites include skyscrapers such as the Empire
State Building and sprawling Central Park. Broadway theater is staged in
neon-lit Times Square.'
  },
  {
    header: { 'text': 'Los Angeles' },
    content: 'Los Angeles is a sprawling Southern California city and the
center of the nation's film and television industry. Near its iconic
Hollywood sign, studios such as Paramount Pictures, Universal and Warner
Brothers offer behind-the-scenes tours. On Hollywood Boulevard, TCL Chinese
Theatre displays celebrities' hand- and footprints, the Walk of Fame honors
thousands of luminaries and vendors sell maps to stars' homes.'
  },
  {
    header: { 'text': 'Chicago' },
    content: 'Chicago, on Lake Michigan in Illinois, is among the largest
cities in the U.S. Famed for its bold architecture, it has a skyline
punctuated by skyscrapers such as the iconic John Hancock Center, 1,451-ft.
Willis Tower (formerly the Sears Tower) and the neo-Gothic Tribune Tower.
The city is also renowned for its museums, including the Art Institute of
Chicago with its noted Impressionist and Post-Impressionist works.'
  }
]
    }, '#usa_tab');
  } else if (e.selectedIndex === 1 &&
isNOU(document.querySelector('#france_tab.e-tab'))) {
    var france_obj= new Tab({
       items:[
  {
    header: { 'text': 'Paris' },
    content: 'Paris, France capital, is a major European city and a global
center for art, fashion, gastronomy and culture. Its 19th-century cityscape
is crisscrossed by wide boulevards and the River Seine. Beyond such
landmarks as the Eiffel Tower and the 12th-century, Gothic Notre-Dame
cathedral, the city is known for its cafe culture and designer boutiques
along the Rue du Faubourg Saint-Honoré.'
```

```
  },
  {
    header: { 'text': 'Marseille' },
    content: 'Marseille, a port city in southern France, has been a
crossroads of immigration and trade since its founding by the Greeks circa
600 B.C. At its heart is the Vieux-Port (Old Port), where fishmongers sell
their catch along the boat-lined quay. Basilique Notre-Dame-de-la-Garde is a
Romanesque-Byzantine church. Modern landmarks include Le Corbusier's
influential Cité Radieuse complex and Zaha Hadid's CMA CGM Tower.'
  },
  {
    header: { 'text': 'Lyon' },
    content: 'Lyon, the capital city in France's Auvergne-Rhône-Alpes
region, sits at the junction of the Rhône and Saône rivers. Its center
reflects 2,000 years of history from the Roman Amphithéâtre des Trois
Gaules, medieval and Renaissance architecture in Vieux (Old) Lyon, to the
modern Confluence district on Presquîle peninsula. Traboules, covered
passageways between buildings, connect Vieux Lyon and La Croix-Rousse hill.'
  }
]
    }, '#france_tab');
  } else if (e.selectedIndex === 2 &&
isNOU(document.querySelector('#australia_tab.e-tab'))) {
    var australia_obj= new Tab({
       items:[
  {
    header: { 'text': 'Sydney' },
    content: 'Sydney, capital of New South Wales and one of Australia
largest cities, is best known for its harbourfront Sydney Opera House, with
a distinctive sail-like design. Massive Darling Harbour and the smaller
Circular Quay port are hubs of waterside life, with the arched Harbour
Bridge and esteemed Royal Botanic Garden nearby. Sydney Tower's outdoor
platform, the Skywalk, offers 360-degree views of the city and suburbs.'
  },
  {
    header: { 'text': 'Melbourne' },
    content: 'Melbourne is the coastal capital of the southeastern
Australian state of Victoria. At the city centre is the modern Federation
Square development, with plazas, bars, and restaurants by the Yarra River.
In the Southbank area, the Melbourne Arts Precinct is the site of Arts
Centre Melbourne - a performing arts complex - and the National Gallery of
Victoria, with Australian and indigenous art.'
  },
  {
    header: { 'text': 'Brisbane' },
    content: 'Brisbane, capital of Queensland, is a large city on the
Brisbane River. Clustered in its South Bank cultural precinct are the
Queensland Museum and Sciencentre, with noted interactive exhibitions.
Another South Bank cultural institution is Queensland Gallery of Modern Art,
among Australia major contemporary art museums. Looming over the city is Mt.
Coot-tha, site of Brisbane Botanic Gardens.'
  }
]
    }, '#australia_tab');
}
 }
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/nested-tab-cs1" %}

Set state persistence of the tab component in Vue Tab component

State persistence allows the Tab to retain the current modal value in the browser cookies for state maintenance. This action is handled through the enablePersistence property which is set to false by default.

When it is set to true, some of the Tab component model values will be retained even after refreshing the page.

The following sample demonstrates how to set state persistence of the Tab component.

APP.VUE

```
<template>
    <div id="app">
    <ejs-tab id='element' enablePersistence='true'>
        <e-tabitems>
                    <e-tabitem :header='headerText0' :content="content0"></e-
tabitem>
                     <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                     <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
      </e-tabitems>
    </ejs-tab>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(TabPlugin);
export default {
  name: 'app',
   data: function(){
        return {
          headerText0: { text: 'ASP.NET' },
          headerText1: { text: 'ASP.NET MVC' },
          headerText2: { text: 'JavaScript' },
        content0: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
```

```
        'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
         content1: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content2: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
        }
    }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/persistence-cs1" %}

Set custom animation in Vue Tab component

Tab supports custom animations for both previous and next actions from the provided animation option of Animation library.

The animation property also allows you to set easing, duration, and various other effects.

Default animation is given as SlideLeftIn for previous tab animation and SlideRightIn for next tab animation. You can also disable the animation by setting the animation effect as None. Also, please use the following CSS to disable indicator animation for animation effect as None.

`css

.e-tab .e-tab-header:not(.e-vertical) .e-indicator, .e-tab .e-tab-header.e-vertical .e-indicator {

transition: none;

}

`

The sample demonstrates some types of animation that suits Tab. You can check all the animation effects here.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="app">
   <div style='padding-top: 25px'>
    <div class='row'>
    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
      <label> Previous Animation </label></div>
      <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
      <div class='custom_drop'><ejs-dropdownlist ref="previousInstance"
:change= 'previousAnimationChange' index='0' :dataSource='animationData'
placeholder='Previous Animation'></ejs-dropdownlist></div>
    </div></div>
    <div class='row'>
    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
    <label> Next Animation </label></div>
    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
    <div class='custom_drop'><ejs-dropdownlist ref="nextInstance" :change=
'nextAnimationChange' index='1' :dataSource='animationData'
placeholder='Next Animation'></ejs-dropdownlist></div>
    </div></div>
    <div style='padding-top: 25px'>
        <ejs-tab id='element' ref="element">
        <e-tabitems>
                  <e-tabitem :header='headerText0' :content="content0"></e-
tabitem>
                   <e-tabitem :header='headerText1'
:content="content1"></e-tabitem>
                   <e-tabitem :header='headerText2'
:content="content2"></e-tabitem>
      </e-tabitems>
    </ejs-tab>
    </div></div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(TabPlugin);
Vue.use(DropDownListPlugin);
export default {
  name: 'app',
 data: function() {
    return {
        headerText0: { text: 'ASP.NET' },
          headerText1: { text: 'ASP.NET MVC' },
          headerText2: { text: 'JavaScript' },
        content0: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
         content1: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content2: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
 animationData:['SlideLeftIn', 'SlideRightIn', 'FadeIn', 'FadeOut',
'FadeZoomIn', 'FadeZoomOut', 'ZoomIn', 'ZoomOut', 'None'];
    };
  },
methods: {
 previousAnimationChange: function(e) {
   var tabObj = this.$refs.element.ej2Instances;
   var ddlobj = this.$refs.previousInstance.ej2Instances;
  tabObj.animation.previous.effect = ddlobj.value;
  }
 nextAnimationChange: function(e) {
      var tabObj = this.$refs.element.ej2Instances;
      var ddlobj = this.$refs.nextInstance.ej2Instances;
      tabObj.animation.next.effect = ddlobj.value;
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/custom-animation-cs1" %}

Load tab items dynamically in Vue Tab component

Tabs can be added dynamically by passing array of items and index value to the addTab method.

`ts

// New tab title and content inputs are fetched and stored in local variable

let title = document.getElementById('tab-title').value;

let content = document.getElementById('tab-content').value;

var tabObj = this.$refs.TabInstance.ej2Instances;

Copyright © 2001 -2024 Syncfusion Inc.

// Required tab item object formed by using textbox inputs

let item = { header: { text: title }, content: createElement('pre', { innerHTML: content.replace(/\n/g, '
\n') }).outerHTML };

// Item object and the index argument passed into the addTab method to add a new tab

tabObj.addTab([item], totalItems-1);

`

In the following demo, you can add the tab content by clicking the **+**. Enter the new Tab heading and content details in the available text boxes, click 'Add Tab' button to pass the details as an array and here last index is calculated to append the new tab at the end.

APP.VUE

```
<template>
    <div id="app">
        <div id='tab1_content' style='display: none'>
            <ul>
                <li>Click on the "+" header to add dynamic tab items. </li>
                <li>It displays input elements to get the new tab
information. </li>
                <li>Add details and click the "Add Tab" button to open the
newly added tab.</li>
            </ul>
        </div>
        <div id='form-container' style='display: none'>
            <div class='e-float-input'>
                <input type='text' id='tab-title' required="" />
                <span class='e-float-line'></span>
                <label class='e-float-text'>Enter header title</label>
            </div>
            <br />
            <div class='e-float-input'>
                <textarea rows='5' type='text' id='tab-content'
required=""></textarea>
                <span class='e-float-line'></span>
                <label class='e-float-text'>Enter content</label>
            </div>
            <br />
            <div class='btn-section'>
                <button id='btn-add' class='e-btn' v-
on:click='btnClicked'>Add Tab</button>
                <br />
                <br />
                <span class='info'> * Title is mandatory to add a new
Tab</span>
            </div>
        </div>
        <ejs-tab id='dynamic_tab' ref='TabInstance'  :selected=
"tabSelected">
        <e-tabitems>
            <e-tabitem :header='headerText0' :content="content0"></e-
tabitem>
            <e-tabitem :header='headerText1' :content="content1"></e-
tabitem>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <e-tabitems>
        </ejs-tab>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { Tab, TabComponent, SelectEventArgs } from '@syncfusion/ej2-
navigations';
import { enableRipple, createElement } from '@syncfusion/ej2-base';
Vue.use(TabPlugin);
export default {
   name: 'app',
   data: function(){
        return {
          totalItems: 0,
          headerText0: { 'text': 'Tab1' },
          headerText1: { 'iconCss': 'e-add-icon' },
         content0: '#tab1_content',
         content1: '#form-container',
        }
  },
  mounted: function() {
    let addBtn = document.querySelectorAll(".e-ileft.e-icon");
    addBtn[0].setAttribute("title", "Add Tab");
  },
  methods:{
  tabSelected: function(args) {
    if (args.selectedIndex === document.querySelectorAll('#dynamic_tab .e-
toolbar-item').length - 1) {
        document.getElementById('tab-title').value = '';
        document.getElementById('tab-content').value = '';
    }
 },
  btnClicked: function(e) {
    let title = document.getElementById('tab-title').value;
    let content = document.getElementById('tab-content').value;
    var tabObj = this.$refs.TabInstance.ej2Instances;
    let item =  { header: { text: title }, content: createElement('pre', {
innerHTML: content.replace(/\n/g, '<br>\n') }).outerHTML };
    let totalItems = document.querySelectorAll('#dynamic_tab .e-toolbar-
item').length;
    tabObj.addTab([item], totalItems-1);
  }
 }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
#container {
    visibility: hidden;
}
#loader {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-content .e-item {
    font-size: 12px;
    margin: 10px;
    text-align: justify;
}
.container {
    min-width: 350px;
    max-width: 500px;
    margin: 0 auto;
}
#form-container {
    margin: 0 auto;
    max-width: 300px;
}
.btn-section {
    text-align: center;
}
.add-tab-btn-section td {
    padding: 10px;
}
.info {
    font-weight: bold;
}
.e-add-icon::before {
    content: '\e823';
}
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/dynamic-tab-cs1" %}

Create collapsible tabs in Vue Tab component

You can achieve collapse and expand functionality in Tab by adding/removing a custom CSS class in the click event handler for each tab.

- Define a CSS class to set the style property display as none. Here 'collapse' class is added to the content element for hiding it.
- Bind the select event for Tab to collapse the initially selected Tab item and bind custom click handler for the Tab headers.
- In the event handler, add and remove 'collapse' class to hide and show the corresponding Tab content.

APP.VUE

```
<template>
  <div id="app">
    <div class="info">
      Collapsible Tabs
```

```
    </div>
    <span style="margin: 10px;">
      <i>The active tab can be toggled to expand and collapse its
content.</i>
    </span>
    <br /><br />
    <ejs-tab id='collapsible_tab' class='e-background' ref='TabInstance'
:created="tabCreated" :selected="tabSelected">
      <e-tabitems>
        <e-tabitem :header='headerText0' :content="content0"></e-tabitem>
        <e-tabitem :header='headerText1' :content="content1"></e-tabitem>
        <e-tabitem :header='headerText2' :content="content2"></e-tabitem>
        <e-tabitems>
    </ejs-tab>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { Tab, TabComponent, SelectEventArgs } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
Vue.use(TabPlugin);
export default {
  name: 'app',
  data: function () {
    return {
      headerText0: { 'text': 'Twitter' },
      headerText1: { 'text': 'Facebook' },
      headerText2: { 'text': 'WhatsApp' },
      content0: 'Twitter is an online social networking service that enables
users to send and read short'
        + '140-character messages called "tweets". Registered users can read
and post tweets, but those'
        + 'who are unregistered can only read them. Users access Twitter
through the website interface,'
        + 'SMS or mobile device app Twitter Inc. is based in San Francisco
and has more than 25 offices'
        + 'around the world. Twitter was created in March 2006 by Jack
Dorsey, Evan Williams, Biz Stone,'
        + 'and Noah Glass and launched in July 2006. The service rapidly
gained worldwide popularity,'
        + 'with more than 100 million users posting 340 million tweets a day
in 2012.The service also'
        + 'handled 1.6 billion search queries per day.',
      content1: 'Facebook is an online social networking service
headquartered in Menlo Park, California.'
        + 'Its website was launched on February 4, 2004, by Mark Zuckerberg
with his Harvard College'
        + 'roommates and fellow students Eduardo Saverin, Andrew McCollum,
Dustin Moskovitz and Chris'
        + 'Hughes.The founders had initially limited the website\'\s
membership to Harvard students, but'
        + 'later expanded it to colleges in the Boston area, the Ivy League,
and Stanford University. It'
        + 'gradually added support for students at various other
universities and later to high-school'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        + 'students.',
      content2: 'WhatsApp Messenger is a proprietary cross-platform instant
messaging client for'
        + 'smartphones that operates under a subscription business model. It
uses the Internet to send'
        + 'text messages, images, video, user location and audio media
messages to other users using'
        + 'standard cellular mobile numbers. As of February 2016, WhatsApp
had a user base of up to one'
        + 'billion,[10] making it the most globally popular messaging
application. WhatsApp Inc., based in'
        + 'Mountain View, California, was acquired by Facebook Inc. on
February 19, 2014, for '
        + 'approximately US$19.3 billion.',
    }
  },
  mounted: function () {
    this.trgIndex;
    this.actLine;
  },
  methods: {
    tabCreated: function () {
      // After tab created first tab content and active line are hidden by
adding custom class to make it collapse state
      this.actLine = document.querySelector('.e-indicator');
      document.getElementById("e-content-
collapsible_tab_0").classList.add('collapse');
      this.actLine.classList.add('collapse');
    },
    tabSelected: function (e) {
      // If next tab item selected custom class is removed from content and
active line element
      let cnttrgs = document.querySelectorAll('#collapsible_tab.e-tab > .e-
content > .e-item');
      for (var i = 0; i < cnttrgs.length; i++) {
        cnttrgs[i].classList.remove('collapse');
      }
      if (this.actLine !== undefined) {
this.actLine.classList.remove('collapse'); }
      this.trgIndex = e.selectedIndex;
      // Custom click event binding for each tab item to make
collapse/expand
      e.selectedItem.addEventListener('click', function (e) {
        this.updateCollapseClass(this.trgIndex);
      }.bind(this));
    },
    updateCollapseClass: function (index) {
      // Custom classes are added/removed from tab content and active line
element, when the same tab item again clicked
      let cntEle = document.getElementById("e-content-collapsible_tab_" +
index);
      if (cntEle.classList.contains('collapse')) {
        cntEle.classList.remove('collapse');
        this.actLine.classList.remove('collapse');
      } else {
        cntEle.classList.add('collapse');
        this.actLine.classList.add('collapse');
```

```
      }
    },
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.e-content .e-item {
  font-size: 12px;
  margin: 10px;
  text-align: justify;
}
.container {
  min-width: 350px;
  margin: 0 10px;
}
.info {
  margin: 10px;
  font-weight: bold;
}
.e-tab .e-content>.e-item.e-active.collapse {
  display: none;
}
.e-tab .e-tab-header .e-indicator.collapse {
  display: none;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/collapsible-tab-cs1" %}

Render other components in tab using template in Vue Tab component

You can render other components inside Tab using Vue **template**. Through this, we can add content as other components directly with all functionalities to our Tab. Follow the below guidelines for using the other components as template in tab.

- Declare a template in the **template** section of the ".vue" file. An empty object "data" needs to be initialized in the data option of the default export object in **script** section.
- The template function needs to be assigned to the content property of the EJ2 Vue Tab Component.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div id="app">
    <ejs-tab id="element" ref=element>
                <e-tabitems>
                    <e-tabitem :header='headerText0'
:content='Template1'></e-tabitem>
                    <e-tabitem :header='headerText1'
:content='Template2'></e-tabitem>
                </e-tabitems>
            </ejs-tab>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { DatePickerComponent,CalendarComponent,
DatePickerPlugin,CalendarPlugin } from '@syncfusion/ej2-vue-calendars';
Vue.use(TabPlugin);
Vue.use(TabPlugin);
Vue.component(DatePickerPlugin.name, DatePickerComponent);
Vue.component(CalendarPlugin.name, CalendarComponent);
export default {
  name: 'app',
  data: function(){
    return {
        headerText0: { text: 'Tab1' },
        headerText1: { text: 'Tab2' },
        headerText2: { text: 'Tab3' },
        Template1: function () {
        return {
          template: Vue.component('DatePickerComponent', {
            template: ' <ejs-datepicker :min="minDate" :max="maxDate"
:value="dateVal" ></ejs-datepicker>',
            data() { return { minDate : new Date("05/09/2017"),
                              maxDate : new Date("05/15/2017"),
                              dateVal : new Date("05/11/2017") }; }
          })
        }
      },
      Template2: function () {
        return {
          template: Vue.component('DatePickerComponent', {
            template: '<ejs-calendar id="calendar" ></ejs-calendar>',
            data() { return { }; }
          })
        }
      }
   }
 }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/direct-components-cs1" %}

Adding dynamic items with content reuse in Vue Tab component

You can add dynamic tabs by reusing the content using Vue **template**. Tabs can be added dynamically by passing array of items and index value to the addTab method.

Content reuse can be achieved by using the following steps:

1. Declare a template in the **template** section of the ".vue" file. An empty object "data" needs to be initialized in the data option of the default export object in **script** section.
2. The template function needs to be assigned to the content property of the EJ2 Vue Tab Component.
3. Provide separate template function for each vue component

and pass content by dynamically adding tab. It will reuse the content of vue component.

Refer to the following sample.

APP.VUE

```
<template>
    <div id="app">
    <button id="add" class="e-btn" v-on:click="addButtonClicked">Click to
add</button>
    <button id="remove" class="e-btn" v-on:click="removeButtonClicked">Click
to remove</button>
    <ejs-tab id="tabElement" ref=tabObj>
                <e-tabitems>
                    <e-tabitem :header='DatePickerHeaderText'
:content='DatePickerTemplate'></e-tabitem>
                    <e-tabitem :header='DropdownHeaderText'
:content='DropDownTemplate'></e-tabitem>
                    <e-tabitem :header='ReusedDropdownHeaderText'
:content='DropDownTemplate'></e-tabitem>
                </e-tabitems>
            </ejs-tab>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { DatePickerComponent, DatePickerPlugin } from '@syncfusion/ej2-vue-
calendars';
import { DropDownListComponent, DropDownListPlugin } from "@syncfusion/ej2-
vue-dropdowns";
Vue.use(TabPlugin);
Vue.component(DatePickerPlugin.name, DatePickerComponent);
Vue.component(DropDownListPlugin.name, DropDownListComponent);
export default {
  name: 'app',
  data: function () {
```

```
    return {
      DatePickerHeaderText: { text: "DatePicker" },
      DropdownHeaderText: { text: "Dropdown" },
      ReusedDropdownHeaderText: { text: "Reused Dropdown" },
      DatePickerTemplate: function () {
        return {
          template: Vue.component("DatePickerComponent", {
            template:
              "<div><h1>{{target}} Content</h1><br /><ejs-datepicker></ejs-
datepicker></div>",
            data() {
              return {
                target: document.querySelector(
                  ".e-toolbar-item.e-active .e-tab-text"
                ).innerHTML,
              };
            },
          }),
        };
      },
      DropDownTemplate: function () {
        return {
          template: Vue.component("DropDownListComponent", {
            template:
              "<div><h1>{{target}} Content</h1><br /><ejs-dropdownlist
height='200px' :dataSource='sportsData' placeholder='Select a game'></ejs-
dropdownlist></div>",
            data() {
              return {
                sportsData: [
                  "Badminton",
                  "Basketball",
                  "Cricket",
                  "Golf",
                  "Hockey",
                  "Rugby",
                ],
                target: document.querySelector(
                  ".e-toolbar-item.e-active .e-tab-text"
                ).innerHTML,
              };
            },
          }),
        };
      },
    };
  },
  methods: {
    addButtonClicked: function (e) {
      var tabObj = this.$refs.tabObj.ej2Instances;
      let newTabItem = {
        header: { text: "DynamicTabItem" },
        content: this.DropDownTemplate,
      };
      tabObj.addTab([newTabItem], 1);
    },
    removeButtonClicked: function (e) {
```

```
      var tabObj = this.$refs.tabObj.ej2Instances;
      tabObj.removeTab(1);
    },
  },
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/how-to/content-reuse-cs1" %}

Customize tab content height in Vue Tab component

You can change the Tab content height by using the heightAdjustMode property. By default, the Tab content heightAdjustMode property is set to Content value.

- **None**: Each tab content height is set based on the Tab height. This value is used only the tab component having the height property.
- **Auto**: Each tab content height will take the maximum height of all other tabs content.
- **Content**: Each tab content height is set based on their own content.
- **Fill**: Each tab content height is set based on the full height of Tabs parent element.

APP.VUE

```
<template>
    <div id="app">
   <div style='padding-top: 25px'>
    <div class='row'>
    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
      <label>Height Adjust Mode</label></div>
      <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
      <div class='custom_drop'><ejs-dropdownlist id='contentHeight'
ref="contentHeight" :change= 'onChange' index='1'
:dataSource='heightData'></ejs-dropdownlist></div>
    </div></div>
    <div style='padding-top: 25px'>
        <ejs-tab id='element' ref="element">
        <e-tabitems>
            <e-tabitem :header='headerText0' :content="content0"></e-
tabitem>
            <e-tabitem :header='headerText1' :content="content1"></e-
tabitem>
            <e-tabitem :header='headerText2' :content="content2"></e-
tabitem>
      </e-tabitems>
    </ejs-tab>
    </div></div>
  </div>
</template>
<script>
```

```
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(TabPlugin);
Vue.use(DropDownListPlugin);
export default {
  name: 'app',
 data: function() {
    return {
        headerText0: { text: 'ASP.NET' },
          headerText1: { text: 'ASP.NET MVC' },
          headerText2: { text: 'JavaScript' },
        content0: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
         content1: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content2: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
 heightData:['None', 'Content', 'Fill', 'Auto'];
    };
  },
methods: {
 onChange: function(e) {
   var tabObj = this.$refs.element.ej2Instances;
   var ddlobj = this.$refs.contentHeight.ej2Instances;
   tabObj.heightAdjustMode = ddlobj.value;
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/tab/how-to/height-cs1" %}

Reorder active tab in Vue Tab component

We can able to prevent the changing of the active tab item on resizing the browser when overflow mode is popup by using the reorderActiveTab property. By default, the active Tab should be reordered when we click the tab items from the popup. If we set false to reorderActiveTab property the active tab item from the popup will not be reordered and an active item is highlighted inside the popup. The following code example depicts to prevent the reorder active tab item inside the popup.

APP.VUE

```
<template>
<div id='container'>
    <ejs-tab id='element' ref="element" heightAdjustMode='Auto'
overflowMode='Popup' reorderActiveTab='false' height='150px' width='700px'>
        <e-tabitems>
            <e-tabitem :header='headerText0' :content="content0"></e-
tabitem>
            <e-tabitem :header='headerText1' :content="content1"></e-
tabitem>
            <e-tabitem :header='headerText2' :content="content2"></e-
tabitem>
            <e-tabitem :header='headerText3' :content="content3"></e-
tabitem>
            <e-tabitem :header='headerText4' :content="content4"></e-
tabitem>
            <e-tabitem :header='headerText5' :content="content5"></e-
tabitem>
            <e-tabitem :header='headerText6' :content="content6"></e-
tabitem>
            <e-tabitem :header='headerText7' :content="content7"></e-
tabitem>
            <e-tabitem :header='headerText8' :content="content8"></e-
tabitem>
            <e-tabitem :header='headerText9' :content="content9"></e-
tabitem>
            <e-tabitem :header='headerText10' :content="content10"></e-
tabitem>
            <e-tabitem :header='headerText11' :content="content11"></e-
tabitem>
       </e-tabitems>
    </ejs-tab>
 </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(TabPlugin);
export default {
  name: 'app',
     data: function(){
        return {
          headerText0: { text: 'HTML' },
          headerText1: { text: 'C Sharp(C#)' },
          headerText2: { text: 'Java' },
```

```
            headerText3: { text: 'VB.Net' },
            headerText4: { text: 'Xamarin' },
            headerText5: { text: 'ASP.NET' },
            headerText6: { text: 'ASP.NET MVC' },
            headerText7: { text: 'JavaScript' },
            headerText8: { text: 'React' },
            headerText9: { text: 'Angular' },
            headerText10: { text: 'Vue' },
            headerText11: { text: 'Typescript' },
            content0: 'HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to ' +
          'create web pages. Along with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually ' +
          'engaging web pages, user interfaces for web applications, and user
interfaces for many mobile applications.[1] Web ' +
          'browsers can read HTML files and render them into visible or
audible web pages. HTML describes the structure of a ' +
          'website semantically along with cues for presentation, making it a
markup language, rather than a programming language.',
           content1: 'C# is intended to be a simple, modern, general-purpose,
object-oriented programming language. Its ' +
          'development team is led by Anders Hejlsberg. The most recent
version is C# 5.0, which was released on August 15, 2012.',
          content2: 'Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired ' +
          'by Oracle Corporation, that provides a system for developing
application software and deploying it in a cross-platform ' +
          'computing environment. Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to ' +
          'enterprise servers and supercomputers. While less common, Java
applets run in secure, sandboxed environments to ' +
          'provide many features of native applications and can be embedded in
HTML pages.',
          content3: 'The command-line compiler, VBC.EXE, is installed as part
of the freeware .NET Framework SDK. Mono also ' +
          'includes a command-line VB.NET compiler. The most recent version is
VB 2012, which was released on August 15, 2012.',
          content4: 'Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers ' +
          'that created Mono,[4] Mono for Android and MonoTouch that are
cross-platform implementations of the Common Language ' +
          'Infrastructure (CLI) and Common Language Specifications (often
called Microsoft .NET). With a C#-shared codebase, ' +
          'developers can use Xamarin tools to write native Android, iOS, and
Windows apps with native user interfaces and share ' +
          'code across multiple platforms.[5] Xamarin has over 1 million
developers in more than 120 countries around the World ' +
          'as of May 2015.',
          content5: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
          'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
          'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
          'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
```

```
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
         content6: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content7: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
        content8:'React is a free and open-source front-end JavaScript
library for building user interfaces based on UI components. It is
maintained by Meta and a community of individual developers and companies.',
        content9:'Angular is a TypeScript-based free and open-source web
application framework led by the Angular Team at Google and by a community
of individuals and corporations. Angular is a complete rewrite from the same
team that built AngularJS',
        content10:'Vue.js is an open-source model-view-viewmodel front end
JavaScript framework for building user interfaces and single-page
applications. It was created by Evan You, and is maintained by him and the
rest of the active core team members.',
        content11:'TypeScript is a programming language developed and
maintained by Microsoft. It is a strict syntactical superset of JavaScript
and adds optional static typing to the language. TypeScript is designed for
the development of large applications and transcompiles to JavaScript.'
        }
        }
    }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  #app {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }

  .e-content .e-item {
    font-size: 12px;
    margin: 10px;
    text-align: justify;
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  #element {
    margin-top: 80px;
    margin-left: 80px;
  }

  #default {
    margin-top: 15px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/orientation-tab-cs1" %}

Tab selection in Vue Tab component

We can able to find the tab selection whether it is selected by user interaction or programmatically way in the selecting and selected event argument with the field of isInteracted. When the user changes the tab through click actions it will return true otherwise, it will return false. The following code example depicts to find the tab selecting the state in selecting and selected events.

APP.VUE

```
<template>
<div id='container'>
    <div className="EventLog" id="EventLog" ></div>
    <div>
      <ejs-dropdownlist id="dropdown" :dataSource='dropdownData'
:change='dropdownChange' index='0' floatLabelType="Always"
placeholder="Select Tab Item using dropdown" :fields='fields' ></ejs-
dropdownlist>
    </div>
    <ejs-tab id='element' ref="element" heightAdjustMode='Auto'
:selecting='selecting' :selected='selected' height='150px' width='700px'>
        <e-tabitems>
            <e-tabitem :header='headerText0' :content="content0"></e-
tabitem>
            <e-tabitem :header='headerText1' :content="content1"></e-
tabitem>
            <e-tabitem :header='headerText2' :content="content2"></e-
tabitem>
            <e-tabitem :header='headerText3' :content="content3"></e-
tabitem>
            <e-tabitem :header='headerText4' :content="content4"></e-
tabitem>
            <e-tabitem :header='headerText5' :content="content5"></e-
tabitem>
            <e-tabitem :header='headerText6' :content="content6"></e-
tabitem>
            <e-tabitem :header='headerText7' :content="content7"></e-
tabitem>
       </e-tabitems>
    </ejs-tab>
 </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
Vue.use(DropDownListPlugin);
Vue.use(TabPlugin);
export default {
  name: 'app',
     data: function(){
        return {
         index:0,
         dropdownData: [
          { text: 'HTML', Id: 0 },
          { text: 'C-Sharp(C#)', Id: 1 },
          { text: 'Java', Id: 2 },
          { text: 'VB.NET', Id: 3 },
          { text: 'Xamarin', Id: 4 },
          { text: 'ASP.NET', Id: 5 },
          { text: 'ASP.NET MVC', Id: 6 },
          { text: 'JavaScript', Id:7 }
        ],
          fields: { text: 'text', value: 'Id' },
          headerText0: { text: 'HTML' },
          headerText1: { text: 'C Sharp(C#)' },
          headerText2: { text: 'Java' },
          headerText3: { text: 'VB.Net' },
          headerText4: { text: 'Xamarin' },
          headerText5: { text: 'ASP.NET' },
          headerText6: { text: 'ASP.NET MVC' },
          headerText7: { text: 'JavaScript' },
          content0: 'HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to ' +
        'create web pages. Along with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually ' +
        'engaging web pages, user interfaces for web applications, and user
interfaces for many mobile applications.[1] Web ' +
        'browsers can read HTML files and render them into visible or
audible web pages. HTML describes the structure of a ' +
        'website semantically along with cues for presentation, making it a
markup language, rather than a programming language.',
         content1: 'C# is intended to be a simple, modern, general-purpose,
object-oriented programming language. Its ' +
        'development team is led by Anders Hejlsberg. The most recent
version is C# 5.0, which was released on August 15, 2012.',
        content2: 'Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired ' +
        'by Oracle Corporation, that provides a system for developing
application software and deploying it in a cross-platform ' +
        'computing environment. Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to ' +
        'enterprise servers and supercomputers. While less common, Java
applets run in secure, sandboxed environments to ' +
        'provide many features of native applications and can be embedded in
HTML pages.',
        content3: 'The command-line compiler, VBC.EXE, is installed as part
of the freeware .NET Framework SDK. Mono also ' +
        'includes a command-line VB.NET compiler. The most recent version is
VB 2012, which was released on August 15, 2012.',
        content4: 'Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers ' +
```

```
        'that created Mono,[4] Mono for Android and MonoTouch that are
cross-platform implementations of the Common Language ' +
        'Infrastructure (CLI) and Common Language Specifications (often
called Microsoft .NET). With a C#-shared codebase, ' +
        'developers can use Xamarin tools to write native Android, iOS, and
Windows apps with native user interfaces and share ' +
        'code across multiple platforms.[5] Xamarin has over 1 million
developers in more than 120 countries around the World ' +
        'as of May 2015.',
        content5: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
         content6: 'The ASP.NET MVC is a web application framework developed
by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
         content7: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
        }
        },
        methods:{
        selected: function(args) {
          this.getInteractionDetail(args.isInteracted);
        },
        selecting: function(args) {
          this.getInteractionDetail(args.isInteracted);
       },
        dropdownChange: function(args){
          let tabObj=document.getElementById('element');
          tabObj.ej2_instances[0].select(args.value);
        },
         getInteractionDetail:function(interact) {
            let eventlog = interact
            ? 'Tab Item selected by user interaction'
              : 'Tab Item selected by programmatically';
            document.getElementById('EventLog').innerHTML =
document.getElementById(
          'EventLog').innerHTML = '<b>' + eventlog + '</b></span>';
          }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
dropdowns/styles/material.css";
  #app {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }

  .e-content .e-item {
    font-size: 12px;
    margin: 10px;
    text-align: justify;
  }

  #element {
    margin-top: 80px;
    margin-left: 80px;
  }

  #default {
    margin-top: 15px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/orientation-tab-cs2" %}

Tab key navigation in Vue Tab component

The tabIndex property of a Tab item is used to enable tab key navigation for that particular item. When a positive value is assigned to the tabIndex property, it allows the user to switch focus to the next or previous tab item using the Tab or Shift+Tab keys. By default, the user can only switch between tab items using the arrow keys.

If the tabIndex value is set to 0 for all tab items, the tab will switch based on the order of the elements on the page. This means that if the tab items are listed in a specific order on the page, the user will be able to navigate through them using the Tab key in that same order.

To use the tabIndex property, you can assign a positive value to the property of each tab item that you want to enable tab key navigation. For example:

APP.VUE

```
<template>
  <div id="app">
    <ejs-tab id='element'>
      <e-tabitems>
        <e-tabitem :header='headerText0' :content="content0"
:tabIndex="0"></e-tabitem>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <e-tabitem :header='headerText1' :content="content1"
:tabIndex="0"></e-tabitem>
        <e-tabitem :header='headerText2' :content="content2"
:tabIndex="0"></e-tabitem>
      </e-tabitems>
    </ejs-tab>
  </div>
</template>
<script>
import Vue from 'vue';
import { TabPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(TabPlugin);
export default {
  name: 'app',
  data: function () {
    return {
      headerText0: { text: 'ASP.NET' },
      headerText1: { text: 'ASP.NET MVC' },
      headerText2: { text: 'JavaScript' },
      content0: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'\s Active Server Pages (ASP) technology. ASP.NET is
built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.',
      content1: 'The ASP.NET MVC is a web application framework developed by
Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source software,
apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET
Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project is
called ASP.NET vNext.',
      content2: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as ' +
        'part of web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop applications.',
    }
  }
}
</script>
<style>
@import
"https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-
base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/tab/tab-container-cs2" %}

With this code, the user will be able to switch between the tab items using the Tab and Shift+Tab keys, in the order specified by the tabIndex values.

It's important to note that the tabIndex property only affects the ability to navigate between tab items using the Tab key. The user will still be able to use the arrow keys to switch between tab items, regardless of the value of the tabIndex property.

Textbox

Getting Started with the Vue Textbox Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Textbox component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies are required to use the TextBox component in your application.

`js

|-- @syncfusion/ej2-vue-inputs

|-- @syncfusion/ej2-inputs

|-- @syncfusion/ej2-base

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

vue create quickstart

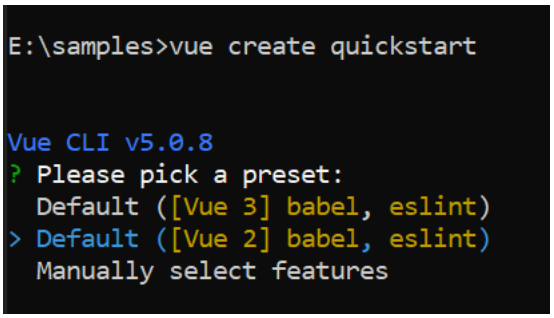
cd quickstart

yarn run serve

Copyright © 2001 -2024 Syncfusion Inc.

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Textbox component as an example. Install the @syncfusion/ej2-vue-inputs package by running the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Textbox component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

The Custom Resource Generator (CRG) is an online web tool, which can be used to generate the custom script and styles for a set of specific components.

Copyright © 2001 -2024 Syncfusion Inc.

This web tool is useful to combine the required component scripts and styles in a single file.

Adding TextBox to the application

Add the input element with e-input class into the <template> section of the App.vue file in src directory.

~/SRC/APP.VUE

```
<template>
<div id ='wrap'>
<div id ='input-container'>
<div>
<!--element which is going to render the TextBox-->
<input class="e-input" type="text" placeholder="Enter Date" />
</div>
</div>
</div>
</template>
```

Adding icons to the TextBox

You can create a TextBox with icon as a group by creating the parent div element with the class e-input-group and add the icon element as span with the class e-input-group-icon. For detailed information, refer to the Groups section.

`

<!--element which is going to render the TextBox with date icon-->

<input class="e-input" name='input' type="text" placeholder="Enter Date"/>

`

`

.e-input-group-icon.e-input-popup-date:before {

content: "\e901";

}

`

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

Copyright © 2001 -2024 Syncfusion Inc.

yarn run serve

`

Output will be as follows:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
                <input class="e-input" type="text" placeholder="Enter Date"
/>
            <div class="e-input-group">
                <input class="e-input" name='input' type="text"
placeholder="Enter Date"/>
                <span class="e-input-group-icon e-input-popup-date"></span>
            </div>
        </div>
    </div>
</template>
<script setup>
import { onMounted } from 'vue';
onMounted(() => {
        // To get the all input fields and its container.
        let inputElement = document.querySelectorAll('.e-input-group .e-
input,.e-float-input.e-input-group input');
        // Add 'e-input-focus' class to the input for achive ripple effect
when focus on the input field.
        for (let i = 0; i < inputElement.length; i++) {
            inputElement[i].addEventListener("focus", function () {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.add('e-input-focus');
                } else {
                    this.parentNode.classList.add('e-input-focus');
                }
            });
            inputElement[i].addEventListener("blur", function () {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.remove('e-input-
focus');
                } else {
                    this.parentNode.classList.remove('e-input-focus');
                }
            });
        }
        // Add 'e-input-btn-ripple' class to the icon element for achive
ripple effect when click on the icon.
        var inputIcon = document.querySelectorAll('.e-input-group-icon');
        for (let i = 0; i < inputIcon.length; i++) {
            inputIcon[i].addEventListener('mousedown', function () {
                this.classList.add('e-input-btn-ripple');
            });
            inputIcon[i].addEventListener('mouseup', function () {
                let element = this;
                setTimeout(function () {
```

```
                    element.classList.remove('e-input-btn-ripple');
                }, 500);
            });
        }
    });
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  #input-container .e-input-group {
    margin: 30px 0;
  }
  .e-input-group-icon:before {
    font-family: e-icons;
  }
  .e-input-group .e-input-group-icon.e-input-popup-date {
    font-size:16px;
  }
  .e-input-group.e-small .e-input-group-icon.e-input-popup-date {
    font-size:14px;
  }
  .e-input-group-icon.e-input-popup-date:before {
    content: "\e901";
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
                <input class="e-input" type="text" placeholder="Enter Date"
/>
            <div class="e-input-group">
                <input class="e-input" name='input' type="text"
placeholder="Enter Date"/>
                <span class="e-input-group-icon e-input-popup-date"></span>
            </div>
        </div>
    </div>
</template>
<script>
export default {
   data: function() {
        return { }
    },
    mounted: function() {
        // To get the all input fields and its container.
        let inputElement = document.querySelectorAll('.e-input-group .e-
input,.e-float-input.e-input-group input');
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        // Add 'e-input-focus' class to the input for achive ripple effect
when focus on the input field.
        for (let i = 0; i < inputElement.length; i++) {
            inputElement[i].addEventListener("focus", function () {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.add('e-input-focus');
                } else {
                    this.parentNode.classList.add('e-input-focus');
                }
            });
            inputElement[i].addEventListener("blur", function () {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.remove('e-input-
focus');
                } else {
                    this.parentNode.classList.remove('e-input-focus');
                }
            });
        }
        // Add 'e-input-btn-ripple' class to the icon element for achive
ripple effect when click on the icon.
        var inputIcon = document.querySelectorAll('.e-input-group-icon');
        for (let i = 0; i < inputIcon.length; i++) {
            inputIcon[i].addEventListener('mousedown', function () {
                this.classList.add('e-input-btn-ripple');
            });
            inputIcon[i].addEventListener('mouseup', function () {
                let element = this;
                setTimeout(function () {
                    element.classList.remove('e-input-btn-ripple');
                }, 500);
            });
        }
    }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  #input-container .e-input-group {
    margin: 30px 0;
  }
  .e-input-group-icon:before {
    font-family: e-icons;
  }
  .e-input-group .e-input-group-icon.e-input-popup-date {
    font-size:16px;
  }
  .e-input-group.e-small .e-input-group-icon.e-input-popup-date {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    font-size:14px;
  }
  .e-input-group-icon.e-input-popup-date:before {
    content: "\e901";
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/icon-samples-cs1" %}

Floating label

The floating label TextBox floats the label above the TextBox after focusing, or filled with value in the TextBox. You can create the floating label TextBox by using `floatLabelType` API.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
            <ejs-textbox id='textbox' floatLabelType="Auto"
placeholder="First Name"></ejs-textbox>
        </div>
    </div>
</template>
<script setup>
import { TextBoxComponent as EjsTextbox } from '@syncfusion/ej2-vue-inputs';
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
            <ejs-textbox id='textbox' floatLabelType="Auto"
placeholder="First Name"></ejs-textbox>
        </div>
    </div>
</template>
<script>
import { TextBoxComponent } from '@syncfusion/ej2-vue-inputs';
export default {
   components: {
    'ejs-textbox': TextBoxComponent
   },
   data: function() {
        return { }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/icon-samples-cs2" %}

See Also

- How to render TextBox programmatically
- How to add floating label to TextBox programmatically

Getting Started with the Vue TextBox Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue TextBox component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

Copyright © 2001 -2024 Syncfusion Inc.

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue TextBox component as an example. To use the Vue TextBox component in the project, the @syncfusion/ej2-vue-inputs package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the TextBox component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue TextBox component using Composition API or Options API:

1.First, import and register the TextBox component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { TextBoxComponent as EjsTextbox } from "@syncfusion/ej2-vue-inputs";
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { TextBoxComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: "App",
components: {
'ejs-textbox' : TextBoxComponent,
}
}
</script>
```

2.In the template section, define the TextBox component with the dataSource property and column definitions.

~/SRC/APP.VUE

```
<template>
<div id ='wrap'>
<div id ='input-container'>
<div>
<!--element which is going to render the TextBox-->
<input class="e-input" type="text" placeholder="Enter Date" />
</div>
</div>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id ='wrap'>
<div id ='input-container'>
<div>
<!--element which is going to render the TextBox-->
<input class="e-input" type="text" placeholder="Enter Date" />
</div>
</div>
</div>
</template>
<script setup>
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
box-sizing: border-box;
margin: 0 auto;
padding: 20px 10px;
width: 340px;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id ='wrap'>
<div id ='input-container'>
<div>
<!--element which is going to render the TextBox-->
<input class="e-input" type="text" placeholder="Enter Date" />
</div>
</div>
</div>
</template>
<script>
export default {
name: 'App',
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
box-sizing: border-box;
margin: 0 auto;
padding: 20px 10px;
width: 340px;
}
</style>
```

Adding icons to the TextBox

You can create a TextBox with icon as a group by creating the parent div element with the class e-input-group and add the icon element as span with the class e-input-group-icon. For detailed information, refer to the Groups section.

~/SRC/APP.VUE

```
<!--element which is going to render the TextBox with date icon-->
<div class="e-input-group">
<input class="e-input" name='input' type="text" placeholder="Enter Date"/>
<span class="e-input-group-icon e-input-popup-date"></span>
</div>
```

~/SRC/APP.VUE

```
.e-input-group-icon.e-input-popup-date:before {
content: "\e901";
}
```

Run the project

To run the project, use the following command:

`bash

Copyright © 2001 -2024 Syncfusion Inc.

npm run dev

`

or

`bash

yarn run dev

`

Floating label

The floating label TextBox floats the label above the TextBox after focusing, or filled with value in the TextBox.

You can create the floating label TextBox by using `floatLabelType` API.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class ='wrap'>
<div id ='input-container'>
<ejs-textbox id='textbox' floatLabelType="Auto" placeholder="First
Name"></ejs-textbox>
</div>
</div>
</template>
<script setup>
import { TextBoxComponent as EjsTextbox } from "@syncfusion/ej2-vue-inputs";
const data = [];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
box-sizing: border-box;
margin: 0 auto;
padding: 20px 10px;
width: 340px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class ='wrap'>
<div id ='input-container'>
<ejs-textbox id='textbox' floatLabelType="Auto" placeholder="First
Name"></ejs-textbox>
</div>
</div>
</template>
<script>
import { TextBoxComponent } from "@syncfusion/ej2-vue-inputs";
export default {
name: 'App',
components: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
"ejs-textbox": TextBoxComponent
},
data: function () {
return {}
},
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
.wrap {
box-sizing: border-box;
margin: 0 auto;
padding: 20px 10px;
width: 340px;
}
</style>
```

The output will appear as follows:

See Also

- How to render TextBox programmatically
- How to add floating label to TextBox programmatically

Groups in Vue Textbox component

The following section explains you the steps required to create TextBox with icon and floating label.

TextBox:

- Create a parent div element with the class e-input-group
- Place input element with the class e-input inside the parent div element.

`

<input class="e-input" name='input' type="text" placeholder="Enter Date"/>

`

Floating label:

- Add the e-float-input class to the parent div element.
- Remove the e-input class and add required attribute to the input element.
- Place the span element with class e-float-line after the input element.
- Place the label element with class e-float-text after the above created span element. When you focus or filled with value in the TextBox, the label floats above the TextBox.

Copyright © 2001 -2024 Syncfusion Inc.

Creating the Floating label TextBox, you have to set the required attribute to the Input element to achieve the floating label functionality which is used for validating the value existence in TextBox. If you want to render the Floating label TextBox without required attribute then refer to the Floating label without required attribute section.

`

<input type="text" required/>

<label class="e-float-text">Enter Name </label>

`

And refer to the following sections to add the icons to the TextBox.

With icon and floating label

Create an icon element as a span with the class e-input-group-icon, and the user can place the icon in either side of TextBox by adding the created icon element before/after the input.

For the floating label enabled TextBox add the icon element as first or last element inside the TextBox wrapper, and based on the element position it will act as prefix or suffix icon.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
            <h4> TextBox with icons </h4>
            <div class="e-input-group">
            <input class="e-input" type="text" placeholder="Enter Date"/>
            <span class="e-input-group-icon e-input-popup-date"></span>
            </div>
            <div class="e-input-group e-float-icon-left">
            <span class="e-input-group-icon e-input-date"></span>
            <div class="e-input-in-wrap">
                <input class="e-input" type="text" placeholder="Enter
Date"/>
            </div>
            </div>
            <div class="e-input-group e-float-icon-left">
            <span class="e-input-group-icon e-input-date"></span>
            <div class="e-input-in-wrap">
                <input class="e-input" type="text" placeholder="Enter
Date"/>
                <span class="e-input-group-icon e-input-down"></span>
            </div>
            </div>
            <h4> Floating label with icons </h4>
            <div class="e-float-input e-input-group">
                <input required type="text" />
                <span class="e-float-line"></span>
                <label class="e-float-text"> Enter Date </label>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <span class="e-input-group-icon e-input-popup-date"></span>
            </div>
            <div class="e-float-input e-input-group e-float-icon-left">
                <span class="e-input-group-icon e-input-date"></span>
                <div class="e-input-in-wrap">
                    <input required type="text" />
                    <span class="e-float-line"></span>
                    <label class="e-float-text"> Enter Date </label>
                </div>
            </div>
            <div class="e-float-input e-input-group e-float-icon-left">
                <span class="e-input-group-icon e-input-date"></span>
                <div class="e-input-in-wrap">
                    <input required type="text" />
                    <span class="e-float-line"></span>
                    <label class="e-float-text"> Enter Date </label>
                    <span class="e-input-group-icon e-input-down"></span>
                </div>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
export default {
   data: function() {
        return { }
    },
    mounted: function() {
        // To get the all input fields and its container.
        let inputElement = document.querySelectorAll('.e-input-group .e-
input,.e-float-input.e-input-group input');
        // Add 'e-input-focus' class to the input for achive ripple effect
when focus on the input field.
        for (let i = 0; i < inputElement.length; i++) {
            inputElement[i].addEventListener("focus", function () {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.add('e-input-focus');
                } else {
                    this.parentNode.classList.add('e-input-focus');
                }
            });
            inputElement[i].addEventListener("blur", function () {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.remove('e-input-
focus');
                } else {
                    this.parentNode.classList.remove('e-input-focus');
                }
            });
        }
        // Add 'e-input-btn-ripple' class to the icon element for achive
ripple effect when click on the icon.
        var inputIcon = document.querySelectorAll('.e-input-group-icon');
        for (let i = 0; i < inputIcon.length; i++) {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            inputIcon[i].addEventListener('mousedown', function () {
                this.classList.add('e-input-btn-ripple');
            });
            inputIcon[i].addEventListener('mouseup', function () {
                let element = this;
                setTimeout(function () {
                    element.classList.remove('e-input-btn-ripple');
                }, 500);
            });
        }
    },
});
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  #input-container .e-input-group { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  #input-container .e-float-input { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  .e-input-group-icon:before {
    font-family: e-icons;
  }
  .e-input-group .e-input-group-icon.e-input-popup-date { /* csslint allow:
adjoining-classes */
    font-size:16px;
  }
  .e-input-group.e-small .e-input-group-icon.e-input-popup-date { /* csslint
allow: adjoining-classes */
    font-size:14px;
  }
  .e-input-group-icon.e-input-popup-date:before { /* csslint allow:
adjoining-classes */
    content: "";
  }
  .e-input-group-icon.e-input-up:before { /* csslint allow: adjoining-
classes */
    content: '\e85e';
  }
  .e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-
classes */
    content: "";
  }
  .e-input-group-icon.e-input-plus:before { /* csslint allow: adjoining-
classes */
    content: '\e7ba';
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  .e-input-group-icon.e-input-minus:before { /* csslint allow: adjoining-
classes */
    content: '\e814';
  }
  .e-input-group-icon.e-input-date:before { /* csslint allow: adjoining-
classes */
    content: "□";
  }
  .e-input-group-icon.e-input-left:before { /* csslint allow: adjoining-
classes */
    content: '\e904';
  }
  .e-input-group-icon.e-input-right:before { /* csslint allow: adjoining-
classes */
    content: '\e913';
  }
  .e-input-group-icon.e-input-reload:before { /* csslint allow: adjoining-
classes */
    content: '\e837';
  }
  .e-input-group-icon.e-input-search:before { /* csslint allow: adjoining-
classes */
    content: '\e806';
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/icon-floating-cs1" %}

To place the icon on left side of the TextBox, create a div element with the class e-input-in-wrap as wrapper to the input element and place the floating line, floating text, and right side icon element within it. Add the e-float-icon-left class to the TextBox container element.

With clear button and floating label

The clear button is added to the input for clearing the value given in the TextBox. It is shown only when the input field has a value, otherwise not shown.

You can add the clear button to the TextBox by using showClearButton API

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
           <h4> TextBox with clear button </h4>
            <ejs-textbox id='textbox' floatLabelType="Never"
showClearButton="true" placeholder="First Name"></ejs-textbox>
            <h4>Floating TextBox with clear button </h4>
            <ejs-textbox id='textbox' floatLabelType="Auto"
showClearButton="true" placeholder="Last Name"></ejs-textbox>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { TextBoxPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(TextBoxPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
 data: function() {
        return { }
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/clear-icon-samples-cs1" %}

Floating label without required attribute

You can render the Floating label TextBox without required attribute by manually float the label above of the TextBox using input events.

You can manually float the label above of the TextBox by adding the below list of classes to the floating label element. The classes are:

Class Name | Description

e-label-top | Floats the label above of the TextBox.

e-label-bottom | Label to be placed as placeholder for the TextBox.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
           <h4> Floating label without required attribute </h4>
            <div class="e-float-input">
                <input type="text"  id='inpt1' />
                <span class="e-float-line"></span>
                <label class="e-float-text"> Enter Value </label>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
export default {
   data: function() {
        return { }
    },
    mounted: function() {
        /* To get the Input element */
        let inputElement = document.getElementById('inpt1');
        /* Update the label position based on Input value */
```

```
        updateLabelState(inputElement.value,
inputElement.parentElement.querySelector('.e-float-text'));
        inputElement.addEventListener("focus", function () {
            let label = this.parentElement.querySelector('.e-float-text');
            label.classList.add('e-label-bottom');
            label.classList.remove('e-label-top');
        });
        inputElement.addEventListener("blur", function () {
            updateLabelState(this.value,
this.parentElement.querySelector('.e-float-text'));
        });
        inputElement.addEventListener("input", function () {
            updateLabelState(this.value,
this.parentElement.querySelector('.e-float-text'));
        });
        /* Update the label position based on Input value */
            /* e-label-top - Floats the label above of the TextBox */
            /* e-label-bottom - Label to be placed as placeholder for the
TextBox */
        function updateLabelState(value,label) {
            if (value) {
                label.classList.add('e-label-top');
                label.classList.remove('e-label-bottom');
            } else {
                label.classList.add('e-label-bottom');
                label.classList.remove('e-label-top');
            }
        }
    }
});
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  #input-container .e-input-group { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  #input-container .e-float-input { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  .e-float-input.e-success label.e-float-text{ /* csslint allow: adjoining-
classes */
    color: #22b24b;
  }
  .e-float-input.e-success input:focus ~ label.e-float-text{ /* csslint
allow: adjoining-classes */
    color: #22b24b;
  }
  .e-float-input.e-success input:valid ~ label.e-float-text { /* csslint
allow: adjoining-classes */
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    color: #22b24b;
  }
  .e-float-input.e-warning label.e-float-text{ /* csslint allow: adjoining-
classes */
    color: #ffca1c;
  }
  .e-float-input.e-warning input:focus ~ label.e-float-text{ /* csslint
allow: adjoining-classes */
    color: #ffca1c;
  }
  .e-float-input.e-warning input:valid ~ label.e-float-text { /* csslint
allow: adjoining-classes */
    color: #ffca1c;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/floating-label-02-cs1" %}

Multi-line input with floating label

Add the HTML textarea element with the e-input class to create default multi-line input.

Add the floating label support to the multi-line input by creating the floating label structure as defined in the initial section.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
           <textarea class="e-input"> Address </textarea>
            <div class="e-float-input">
                <textarea required></textarea>
                <span class="e-float-line"></span>
                <label class="e-float-text"> Address </label>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
export default {
   data: function() {
        return { }
    },
    mounted: function() {
        let input = document.querySelectorAll('.e-input-group .e-input,.e-
float-input.e-input-group input');
        let inputIcon = document.querySelectorAll('.e-input-group-icon');
        let localObj = this;
        for (let i = 0; i < input.length; i++) {
        //Focus Event binding for input component
            input[i].addEventListener('focus', function() {
               localObj.getParentNode(input[i]).classList.add('e-input-
focus');
            });
            //Blur Event binding for input component
```

```
            input[i].addEventListener('blur', function() {
                localObj.getParentNode(input[i]).classList.remove('e-input-
focus');
            });
        }
        for (let i = 0; i < inputIcon.length; i++) {
            inputIcon[i].addEventListener('mousedown',  function() {
                this.classList.add('e-input-btn-ripple');
            });
            inputIcon[i].addEventListener('mouseup',  function() {
                let ele = this;
                setTimeout( function() {
                        ele.classList.remove('e-input-btn-ripple');
                    },
                500);
            });
        }
    },
    methods: {
        getParentNode: function(element) {
            let parentNode = element.parentNode;
            if (parentNode.classList.contains('e-input-in-wrap')) {
                return parentNode.parentNode;
            }
            return parentNode;
        }
    }
});
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  #input-container .e-input-group { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  #input-container .e-float-input { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/getting-started-html-cs1" %}

See Also

- How to add floating label to TextBox programmatically
- Change the floating label color of the TextBox
- Change the color of the TextBox based on its value

Copyright © 2001 -2024 Syncfusion Inc.

Sizing in Vue Textbox component

You can render the TextBox in two different sizes.

Property | Description

Normal | By default, the TextBox is rendered with normal size.

Small | You need to add e-small class to the input element, or else add to the input container.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
            <h4> Normal Size </h4>
            <div class="e-input-group">
                <input class="e-input" type="text" placeholder="Enter Name"
/>
            </div>
            <div class="e-float-input">
                <input type='text' required />
                <span class="e-float-line"></span>
                <label class="e-float-text">Enter Name</label>
            </div>
            <div class="e-input-group">
                <input class="e-input" type="text" placeholder="Enter Date"
/>
                <span class="e-input-group-icon e-input-popup-date"></span>
            </div>
            <h4> Small Size </h4>
            <div class="e-input-group e-small">
                <input class="e-input" type="text" placeholder="Enter Name"
/>
            </div>
            <div class="e-float-input e-small">
                <input type='text' required />
                <span class="e-float-line"></span>
                <label class="e-float-text">Enter Name</label>
            </div>
            <div class="e-input-group e-small">
                <input class="e-input" type="text" placeholder="Enter Date"
/>
                <span class="e-input-group-icon e-input-popup-date"></span>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
export default {
   data: function() {
        return { }
    },
    mounted: function() {
        let input = document.querySelectorAll('.e-input-group .e-input,.e-
float-input.e-input-group input');
        let inputIcon = document.querySelectorAll('.e-input-group-icon');
        let localObj = this;
```

```
        for (let i = 0; i < input.length; i++) {
        //Focus Event binding for input component
            input[i].addEventListener('focus', function() {
               localObj.getParentNode(input[i]).classList.add('e-input-
focus');
            });
            //Blur Event binding for input component
            input[i].addEventListener('blur', function() {
                localObj.getParentNode(input[i]).classList.remove('e-input-
focus');
            });
        }
        for (let i = 0; i < inputIcon.length; i++) {
            inputIcon[i].addEventListener('mousedown',  function() {
                this.classList.add('e-input-btn-ripple');
            });
            inputIcon[i].addEventListener('mouseup',  function() {
                let ele = this;
                setTimeout( function() {
                          ele.classList.remove('e-input-btn-ripple');
                    },
                500);
            });
        }
    },
    methods: {
        getParentNode: function(element) {
            let parentNode = element.parentNode;
            if (parentNode.classList.contains('e-input-in-wrap')) {
                return parentNode.parentNode;
            }
            return parentNode;
        }
    }
});
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  #input-container .e-input-group { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  #input-container .e-float-input { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  .e-input-group-icon:before {
    font-family: e-icons;
  }
  .e-input-group .e-input-group-icon.e-input-popup-date { /* csslint allow:
adjoining-classes */
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    font-size:16px;
  }
  .e-input-group.e-small .e-input-group-icon.e-input-popup-date { /* csslint
allow: adjoining-classes */
    font-size:14px;
  }
  .e-input-group-icon.e-input-popup-date:before { /* csslint allow:
adjoining-classes */
    content: "";
  }
  .e-input-group-icon.e-input-up:before { /* csslint allow: adjoining-
classes */
    content: '\e85e';
  }
  .e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-
classes */
    content: "";
  }
  .e-input-group-icon.e-input-plus:before { /* csslint allow: adjoining-
classes */
    content: '\e7ba';
  }
  .e-input-group-icon.e-input-minus:before { /* csslint allow: adjoining-
classes */
    content: '\e814';
  }
  .e-input-group-icon.e-input-date:before { /* csslint allow: adjoining-
classes */
    content: "";
  }
  .e-input-group-icon.e-input-left:before { /* csslint allow: adjoining-
classes */
    content: '\e904';
  }
  .e-input-group-icon.e-input-right:before { /* csslint allow: adjoining-
classes */
    content: '\e913';
  }
  .e-input-group-icon.e-input-reload:before { /* csslint allow: adjoining-
classes */
    content: '\e837';
  }
  .e-input-group-icon.e-input-search:before { /* csslint allow: adjoining-
classes */
    content: '\e806';
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/icon-samples-cs3" %}

Validation in Vue Textbox component

The TextBox supports three types of validation styles namely error, warning, and success. These states are enabled by adding corresponding classes .e-error, .e-warning, or .e-success to the input parent element.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
            <div class="e-input-group e-warning">
                <input class="e-input" type="text" placeholder="Input with
warning" />
            </div>
            <div class="e-input-group e-error">
                <input class="e-input" type="text" placeholder="Input with
error" />
            </div>
            <div class="e-input-group e-success">
                <input class="e-input" type="text" placeholder="Input with
success" />
            </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
export default {
   data: function() {
        return { }
    },
    mounted: function() {
        let input = document.querySelectorAll('.e-input-group .e-input,.e-
float-input.e-input-group input');
        let inputIcon = document.querySelectorAll('.e-input-group-icon');
        let localObj = this;
        for (let i = 0; i < input.length; i++) {
        //Focus Event binding for input component
            input[i].addEventListener('focus', function() {
               localObj.getParentNode(input[i]).classList.add('e-input-
focus');
            });
            //Blur Event binding for input component
            input[i].addEventListener('blur', function() {
                localObj.getParentNode(input[i]).classList.remove('e-input-
focus');
            });
        }
        for (let i = 0; i < inputIcon.length; i++) {
            inputIcon[i].addEventListener('mousedown',  function() {
                this.classList.add('e-input-btn-ripple');
            });
            inputIcon[i].addEventListener('mouseup',  function() {
                let ele = this;
                setTimeout( function() {
                          ele.classList.remove('e-input-btn-ripple');
                    },
                500);
            });
        }
    },
    methods: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        getParentNode: function(element) {
            let parentNode = element.parentNode;
            if (parentNode.classList.contains('e-input-in-wrap')) {
                return parentNode.parentNode;
            }
            return parentNode;
        }
    }
});
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  #input-container .e-input-group { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  .e-float-input.e-success label.e-float-text{ /* csslint allow: adjoining-
classes */
    color: #22b24b;
  }
  .e-float-input.e-success input:focus ~ label.e-float-text{ /* csslint
allow: adjoining-classes */
    color: #22b24b;
  }
  .e-float-input.e-success input:valid ~ label.e-float-text { /* csslint
allow: adjoining-classes */
    color: #22b24b;
  }
  .e-float-input.e-warning label.e-float-text{ /* csslint allow: adjoining-
classes */
    color: #ffca1c;
  }
  .e-float-input.e-warning input:focus ~ label.e-float-text{ /* csslint
allow: adjoining-classes */
    color: #ffca1c;
  }
  .e-float-input.e-warning input:valid ~ label.e-float-text { /* csslint
allow: adjoining-classes */
    color: #ffca1c;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/icon-samples-cs4" %}

Adding mandatory asterisk to placeholder and float label

You can add a mandatory asterisk(*) to placeholder and float label using .e-float-input.e-control-wrapper .e-float-text::after class.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
            <ejs-textbox id='textbox' floatLabelType="Auto"
placeholder="First Name"></ejs-textbox>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { TextBoxPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(TextBoxPlugin);
export default {
   data: function() {
        return { }
    };
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  .e-float-input.e-control-wrapper .e-float-text::after {
    content: '*';
    color: red;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/icon-samples-cs5" %}

Multiline in Vue Textbox component

This feature allows the textbox to accept one or more lines of text like address, description, comments, and more.

Create multiline textbox

You can convert the default textbox into the multiline textbox by setting the multiline API value as true or pass HTML5 textarea as element to the textbox.

The multiline textbox allows you to resize it in vertical direction alone.

APP.VUE

```
<template>
        <div class="multiline">
            <ejs-textbox name='default' :multiline="true" placeholder="Enter
your address" value= 'Mr. Dodsworth Dodsworth, System Analyst, Studio 103,
The Business Center'></ejs-textbox>
        </div>
    </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { TextBoxPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(TextBoxPlugin);
export default {
 data: function(){
        return {
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .multiline{
    margin: 10px auto;
    width: 30%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/textarea-cs1" %}

Implementing floating label

You can achieve the floating label behavior in the multiline textbox by setting floatLabelType as 'Auto'. The placeholder text act as floating label to the multiline textbox. You can provide the placeholder text to the multiline textbox either by using the placeholder property or placeholder attribute.

APP.VUE

```
<template>
        <div class="multiline">
            <label class="label">Float label type auto</label>
            <ejs-textbox id='default' :multiline="true" placeholder="Enter
your address" floatLabelType="Auto" maxlength="15"></ejs-textbox>
            <label class="label">Float label type always</label>
            <ejs-textbox id='default' :multiline="true" placeholder="Enter
your address" floatLabelType="Always"></ejs-textbox>
            <label class="label">Float label type never</label>
            <ejs-textbox id='default' :multiline="true" placeholder="Enter
your address" floatLabelType="Never"></ejs-textbox>
        </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
</template>
<script>
import Vue from 'vue';
import { TextBoxPlugin } from '@syncfusion/ej2-vue-inputs';
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(TextBoxPlugin);
Vue.use(ButtonPlugin);
export default {
 data: function(){
        return {
             clickHandler: function(event) {
                  this.$refs.textareaObj.addAttributes({maxlength: 15});
             }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .multiline{
    margin: 10px auto;
    width: 30%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/float-cs1" %}

Auto resizing

By default, you can manually resize the multiline textbox. But you can also create an auto resizing multiline textbox with both the initial and dynamic value change. It can be done by calculating the height of the textarea in the created event for initial value update and in the input event for dynamic value update of the auto resize multiline textbox, as explained in the following code sample.

APP.VUE

```
<template>
        <div class="multiline">
            <ejs-textbox ref="textareaObj" id='default' :multiline="true"
value= 'Mr. Dodsworth Dodsworth, System Analyst, Studio 103, The Business
```

```
Center' placeholder="Enter your address" floatLabelType="Auto" :input=
"inputHandler" :created= "createHandler"></ejs-textbox>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { TextBoxPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(TextBoxPlugin);
export default {
 data: function(){
        return {
            createHandler: (args) => {
                this.$refs.textareaObj.addAttributes({rows: 1});
                this.$refs.textareaObj.$el.style.height = "auto";
                this.$refs.textareaObj.$el.style.height =
(this.$refs.textareaObj.$el.scrollHeight)+"px";
            },
            inputHandler: (args) => {
                args.event.currentTarget.style.height = "auto";
                args.event.currentTarget.style.height =
(args.event.currentTarget.scrollHeight)+"px";
            }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .multiline{
        margin: 10px auto;
        width: 30%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/auto-resize-cs1" %}

Disable resizing

By default, the multiline textbox is rendered with resizable. You can disable the resize of the multiline textbox by applying the following CSS styles.

Copyright © 2001 -2024 Syncfusion Inc.

`css

textarea.e-input,

.e-float-input textarea,

.e-float-input.e-control-wrapper textarea,

.e-input-group textarea,

.e-input-group.e-control-wrapper textarea {

resize: none;

}

`

APP.VUE

```
<template>
        <div class="multiline">
            <ejs-textbox name='default' :multiline="true" placeholder="Enter
your address" floatLabelType="Auto" cssClass="sample"></ejs-textbox>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { TextBoxPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(TextBoxPlugin);
export default {
 data: function(){
        return {
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .multiline{
        margin: 10px auto;
        width: 30%;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
    .sample textarea[name=default]{
        resize: none;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/disable-cs1" %}

Limit the text length

By default, the text length of the multiline textbox is unlimited. You can limit the text length by setting the maxLength attribute using the addAttributes method.

APP.VUE

```
<template>
        <div class="multiline">
            <label class="label">Add maxlength attribute through
inline</label>
            <ejs-textbox id='default' :multiline="true" placeholder="Enter
your address" floatLabelType="Auto" maxlength="15"></ejs-textbox>
            <label class="label">Add maxlength attribute through
addAttributes method</label>
            <ejs-textbox id='default' :multiline="true" placeholder="Enter
your address" floatLabelType="Auto" ref="textareaObj"></ejs-textbox>
             <ejs-button id='modalbtn' v-on:click.native="modalBtnClick">Add
max length</ejs-button>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { TextBoxPlugin } from '@syncfusion/ej2-vue-inputs';
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(TextBoxPlugin);
Vue.use(ButtonPlugin);
export default {
 data: function(){
        return {
             modalBtnClick: function() {
                 this.$refs.textareaObj.addAttributes({maxlength: 15});
             }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .multiline{
    margin: 10px auto;
    width: 30%;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/maxlength-cs1" %}

Count characters

You can show the number of characters entered inside the textarea by calculating the character count in the input event of multiline textbox. The character count is updated while entering or deleting any character inside the textarea. The character count shows how many characters can be entered or left to be entered.

APP.VUE

```
<template>
        <div class="multiline">
            <ejs-textbox id='default' :multiline="true" placeholder="Enter
your address" floatLabelType="Auto" :input= "inputHandler"
maxlength="25"></ejs-textbox>
            <span id='numbercount'></span>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { TextBoxPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(TextBoxPlugin);
export default {
 data: function(){
        return {
            inputHandler: (args) => {
                let word, addresscountRem, addressCount;
                word = args.value;
                addressCount = word.length;
                addresscountRem = document.getElementById('numbercount');
                addresscountRem.textContent = addressCount+"/25";
            }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .multiline{
    margin: 10px auto;
    width: 30%;
  }
  #numbercount{
    margin-left: 170px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/auto-resize-cs2" %}

Accessibility in Vue Textbox component

The Textbox component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Textbox component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Textbox is characterized with complete ARIA Accessibility support that helps to access through the on-screen readers and other assistive technology devices. This component is designed with the reference of the guidelines document given in WAI ARAI Accessibility practices.

The Textbox uses the `textbox` role and following ARIA properties for its element based on its state.

| **Property** | **Functionality** |

| --- | --- |

| aria-placeholder | The `aria-placeholder` is a short hint to help the users with data entry when the Textbox has no value. |

| aria-labelledby | The `aria-labelledby` property indicates the floating label element of the Textbox. |

Ensuring accessibility

The Textbox component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Textbox component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Textbox component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/textbox.html" %}

See also

- Accessibility in Syncfusion Vue components

Two way binding in Vue Textbox component

Two-way binding can be achieved by using the `v-model` directive in Vue. In the following sample, when you change the value in one Textbox component, v-model will automatically update the value in the other Textbox.

Copyright © 2001 -2024 Syncfusion Inc.

The following example demonstrates how to set the two-way-binding in the Textbox.

APP.VUE

```
<template>
<div id="app">
    <div id="wrapper1">
        <ejs-textbox floatLabelType="Auto" placeholder="Enter a name" v-
model="value"></ejs-textbox>
    </div>
    <div id="wrapper2">
        <ejs-textbox floatLabelType="Auto" placeholder="Enter a name" v-
model="value"></ejs-textbox>
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { TextBoxPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(TextBoxPlugin);
export default {
 data: function(){
        return {
            value: null
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#wrapper1{
    min-width: 250px;
    float: left;
    margin-left: 100px;
}
#wrapper2{
    min-width: 250px;
    float: right;
    margin-right: 100px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/two-way-cs1" %}

Style appearance in Vue Textbox component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of TextBox wrapper element

Use the following CSS to customize the appearance of wrapper element.

`

/ To specify height and font size /

Copyright © 2001 -2024 Syncfusion Inc.

.e-input:not(:valid), .e-input:valid, .e-float-input.e-control-wrapper input:not(:valid), .e-float-input.e-control-wrapper input:valid, .e-float-input input:not(:valid), .e-float-input input:valid, .e-input-group input:not(:valid), .e-input-group input:valid, .e-input-group.e-control-wrapper input:not(:valid), .e-input-group.e-control-wrapper input:valid, .e-float-input.e-control-wrapper textarea:not(:valid), .e-float-input.e-control-wrapper textarea:valid, .e-float-input textarea:not(:valid), .e-float-input textarea:valid, .e-input-group.e-control-wrapper textarea:not(:valid), .e-input-group.e-control-wrapper textarea:valid, .e-input-group textarea:not(:valid), .e-input-group textarea:valid {

font-size: 30px;

height: 40px;

}

`

Customizing the TextBox placeholder

Use the following CSS to customize the TextBox placeholder

`

/ To specify font size and color /

.e-float-input.e-control-wrapper:not(.e-error) input:valid ~ label.e-float-text, .e-float-input.e-control-wrapper:not(.e-error) input ~ label.e-label-top.e-float-text {

color: pink;

font-size: 15px;

}

`

How To

Set the rounded corner in Vue Textbox component

Render the TextBox with `rounded corner` by adding the `e-corner` class to the input parent element.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
            <div class="e-input-group e-corner">
                <input class="e-input" type="text" placeholder="Enter Date"
/>
                <span class="e-input-group-icon e-input-popup-date"></span>
            </div>
            <div class="e-float-input e-input-group e-corner">
                <input type='text' required />
                <span class="e-float-line"> </span>
                <label class="e-float-text">Enter Date </label>
                <span class="e-input-group-icon e-input-popup-date"></span>
            </div>
        </div>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
export default {
   data: function() {
        return { }
    },
    mounted: function() {
        // To get the all input fields and its container.
        let inputElement = document.querySelectorAll('.e-input-group .e-
input,.e-float-input.e-input-group input');
        // Add 'e-input-focus' class to the input for achive ripple effect
when focus on the input field.
        for (let i = 0; i < inputElement.length; i++) {
            inputElement[i].addEventListener("focus", function () {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.add('e-input-focus');
                } else {
                    this.parentNode.classList.add('e-input-focus');
                }
            });
            inputElement[i].addEventListener("blur", function () {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.remove('e-input-
focus');
                } else {
                    this.parentNode.classList.remove('e-input-focus');
                }
            });
        }
        // Add 'e-input-btn-ripple' class to the icon element for achive
ripple effect when click on the icon.
        var inputIcon = document.querySelectorAll('.e-input-group-icon');
        for (let i = 0; i < inputIcon.length; i++) {
            inputIcon[i].addEventListener('mousedown', function () {
                this.classList.add('e-input-btn-ripple');
            });
            inputIcon[i].addEventListener('mouseup', function () {
                let element = this;
                setTimeout(function () {
                    element.classList.remove('e-input-btn-ripple');
                }, 500);
            });
        }
    });
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
```

```
  #input-container .e-input-group { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  #input-container .e-float-input { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  .e-input-group-icon:before {
    font-family: e-icons;
  }
  .e-input-group .e-input-group-icon.e-input-popup-date { /* csslint allow:
adjoining-classes */
    font-size:16px;
  }
  .e-input-group.e-small .e-input-group-icon.e-input-popup-date { /* csslint
allow: adjoining-classes */
    font-size:14px;
  }
  .e-input-group-icon.e-input-popup-date:before { /* csslint allow:
adjoining-classes */
    content: "";
  }
  .e-input-group-icon.e-input-up:before { /* csslint allow: adjoining-
classes */
    content: '\e85e';
  }
  .e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-
classes */
    content: "";
  }
  .e-input-group-icon.e-input-plus:before { /* csslint allow: adjoining-
classes */
    content: '\e7ba';
  }
  .e-input-group-icon.e-input-minus:before { /* csslint allow: adjoining-
classes */
    content: '\e814';
  }
  .e-input-group-icon.e-input-date:before { /* csslint allow: adjoining-
classes */
    content: "";
  }
  .e-input-group-icon.e-input-left:before { /* csslint allow: adjoining-
classes */
    content: '\e904';
  }
  .e-input-group-icon.e-input-right:before { /* csslint allow: adjoining-
classes */
    content: '\e913';
  }
  .e-input-group-icon.e-input-reload:before { /* csslint allow: adjoining-
classes */
    content: '\e837';
  }
  .e-input-group-icon.e-input-search:before { /* csslint allow: adjoining-
classes */
    content: '\e806';
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/round-corner-cs1" %}

Set the disabled state in Vue Textbox component

Disable the TextBox by adding the e-disabled to the input parent element and set disabled attribute to the input element.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
            <input class="e-input" type="text" placeholder="Enter Name"
disabled/>
            <div class="e-float-input e-disabled">
                <input type='text' required  disabled/>
                <span class="e-float-line"></span>
            <label class="e-float-text">Enter Name</label>
        </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
export default {
   data: function() {
        return { }
    },
    mounted: function() {
        // To get the all input fields and its container.
        let inputElement = document.querySelectorAll('.e-input-group .e-
input,.e-float-input.e-input-group input');
        // Add 'e-input-focus' class to the input for achive ripple effect
when focus on the input field.
        for (let i = 0; i < inputElement.length; i++) {
            inputElement[i].addEventListener('focus', () => {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.add('e-input-focus');
                } else {
                    this.parentNode.classList.add('e-input-focus');
                }
            });
            inputElement[i].addEventListener('blur', () => {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.remove('e-input-
focus');
                } else {
                    this.parentNode.classList.remove('e-input-focus');
                }
            });
        }
    });
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/getting-started-html-cs2" %}

Set the read only textbox in Vue Textbox component

You can make the TextBox as read-only by setting the readonly attribute to the input element.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
            <input class="e-input" type="text" placeholder="Enter Name"
value="John" readonly/>
            <div class="e-float-input">
                <input type='text' required readonly value="John"/>
                <span class="e-float-line"></span>
                <label class="e-float-text e-label-top">Enter Name</label>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
export default {
   data: function() {
        return { }
    },
    mounted: function() {
        // To get the all input fields and its container.
        let inputElement = document.querySelectorAll('.e-input-group .e-
input,.e-float-input.e-input-group input');
        // Add 'e-input-focus' class to the input for achive ripple effect
when focus on the input field.
        for (let i = 0; i < inputElement.length; i++) {
            inputElement[i].addEventListener('focus', () => {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.add('e-input-focus');
                } else {
                    this.parentNode.classList.add('e-input-focus');
                }
            });
            inputElement[i].addEventListener('blur', () => {
                let parentElement = this.parentNode;
```

```
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.remove('e-input-
focus');
                } else {
                    this.parentNode.classList.remove('e-input-focus');
                }
            });
        }
    });
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  #input-container .e-float-input { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/getting-started-html-cs3" %}

Add textbox programmatically in Vue Textbox component

- Create a TypeScript file and import the Input modules from ej2-inputs library as shown below.

`ts

import {Input} from '@syncfusion/ej2-inputs';

`

- Pass the HTML Input element as parameter to the createInput method.
- You can also add the icons on the input by passing buttons property value with the class name e-input-group-icon as parameter to the createInput method.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { Input, InputObject } from  '@syncfusion/ej2-inputs';
export default {
```

```
    data: function() {
        return { }
    },
    mounted: function() {
        // To get the all input fields and its container.
        let inputElement = document.querySelectorAll('.e-input-group .e-
input,.e-float-input.e-input-group input');
        // Add 'e-input-focus' class to the input for achive ripple effect
when focus on the input field.
        for (let i = 0; i < inputElement.length; i++) {
            inputElement[i].addEventListener("focus", function () {
                this.parentNode.classList.add('e-input-focus')
            });
            inputElement[i].addEventListener("blur", function () {
                this.parentNode.classList.remove('e-input-focus')
            });
        }
        // Add 'e-input-btn-ripple' class to the icon element for achive
ripple effect when click on the icon.
        var inputIcon = document.querySelectorAll('.e-input-group-icon');
        for (let i = 0; i < inputIcon.length; i++) {
            inputIcon[i].addEventListener('mousedown', function () {
                this.classList.add('e-input-btn-ripple');
            });
            inputIcon[i].addEventListener('mouseup', function () {
                var element = this;
                setTimeout(function () {
                    element.classList.remove('e-input-btn-ripple');
                }, 500);
            });
        }
        let inputObj;
        let element = document.createElement('input');
        document.getElementById('input-container').appendChild(element);
        Input.createInput({
            element: element,
            properties:{
                placeholder:'Enter Name'
            }
        });
        let element1 = document.createElement('input');
        document.getElementById('input-container').appendChild(element1);
        Input.createInput({
            element: element1,
            buttons: ['e-input-group-icon e-input-down'],
            properties:{
                placeholder:'Enter Value'
            }
        });
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
```

```
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  #input-container .e-input-group { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  .e-input-group .e-input-group-icon.e-input-popup-date { /* csslint allow:
adjoining-classes */
    font-size:16px;
  }
  .e-input-group-icon:before {
    font-family: e-icons;
  }
  .e-input-group-icon.e-input-up:before { /* csslint allow: adjoining-
classes */
    content: '\e85e';
  }
  .e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-
classes */
    content: "";
  }
  .e-input-group-icon.e-input-plus:before { /* csslint allow: adjoining-
classes */
    content: '\e7ba';
  }
  .e-input-group-icon.e-input-minus:before { /* csslint allow: adjoining-
classes */
    content: '\e814';
  }
  .e-input-group-icon.e-input-popup-date:before { /* csslint allow:
adjoining-classes */
    content: "";
  }
  .e-input-group-icon.e-input-date:before { /* csslint allow: adjoining-
classes */
    content: "";
  }
  .e-input-group-icon.e-input-left:before { /* csslint allow: adjoining-
classes */
    content: '\e904';
  }
  .e-input-group-icon.e-input-right:before { /* csslint allow: adjoining-
classes */
    content: '\e913';
  }
  .e-input-group-icon.e-input-reload:before { /* csslint allow: adjoining-
classes */
    content: '\e837';
  }
  .e-input-group-icon.e-input-search:before { /* csslint allow: adjoining-
classes */
    content: '\e806';
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/utility-rendering-cs1" %}

Add floating label to textbox programmatically in Vue Textbox component

The Floating Label TextBox floats label above the TextBox after focusing, or entering a value in the TextBox.

Floating label supports the types of actions as given below.

Type | Description

Auto | The floating label will float above the input after focusing, or entering a value in the input.

Always | The floating label will always float above the input.

Never | By default, never float the label in the input when the placeholder is available.

- Import the Input modules from ej2-inputs library as shown below.

`

import {Input} from '@syncfusion/ej2-inputs';

`

- Pass the HTML Input element and floatLabelType property as Auto to the createInput method.
- Set the placeholder value to the input element via element attribute or pass the parameter to the createInput method.

The watermark label will be updated based on the specified placeholder value of the Floating Label TextBox.

- You can add the icons on the input by passing buttons property value with the class name e-input-group-icon as parameter to the createInput method.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id="input-container">
            <h4> FloatLabelType as Auto </h4>
        </div>
        <div id="input-container-01">
            <h4> FloatLabelType as Always </h4>
        </div>
        <div id="input-container-02">
            <h4> FloatLabelType as Never </h4>
        </div>
        <div id="input-container-03">
            <h4> Float label input with icons </h4>
        </div>
    </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { Input, InputObject } from  '@syncfusion/ej2-inputs';
import { enableRipple } from '@syncfusion/ej2-base';
export default {
   data: function() {
        return { }
    },
    mounted: function() {
        let inputObj;
        let element = document.createElement('input');
        document.getElementById('input-container').appendChild(element);
        Input.createInput({
            element: element,
            floatLabelType: "Auto",
            properties:{
                placeholder:'Enter Name'
            }
        });
        let element2 = document.createElement('input');
        document.getElementById('input-container-01').appendChild(element2);
        Input.createInput({
            element: element2,
            floatLabelType: "Always",
            properties:{
                placeholder:'Enter Name'
            }
        });
        let element3 = document.createElement('input');
        document.getElementById('input-container-02').appendChild(element3);
        Input.createInput({
            element: element3,
            floatLabelType: "Never",
            properties:{
                placeholder:'Enter Name'
            }
        });
        // Render float label TextBox with icon
        let element4 = document.createElement('input');
        document.getElementById('input-container-03').appendChild(element4);
        Input.createInput({
            element: element4,
            floatLabelType: "Auto",
            buttons: ['e-input-group-icon e-input-down'],
            properties:{
                placeholder:'Enter Value'
            }
        });
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
```

```
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  #input-container .e-input-group { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  #input-container .e-float-input { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  .e-input-group-icon:before {
    font-family: e-icons;
  }
  .e-input-group .e-input-group-icon.e-input-popup-date { /* csslint allow:
adjoining-classes */
    font-size:16px;
  }
  .e-input-group-icon.e-input-popup-date:before { /* csslint allow:
adjoining-classes */
    content: "";
  }
  .e-input-group-icon.e-input-up:before { /* csslint allow: adjoining-
classes */
    content: '\e85e';
  }
  .e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-
classes */
    content: "";
  }
  .e-input-group-icon.e-input-plus:before { /* csslint allow: adjoining-
classes */
    content: '\e7ba';
  }
  .e-input-group-icon.e-input-minus:before { /* csslint allow: adjoining-
classes */
    content: '\e814';
  }
  .e-input-group-icon.e-input-date:before { /* csslint allow: adjoining-
classes */
    content: "";
  }
  .e-input-group-icon.e-input-left:before { /* csslint allow: adjoining-
classes */
    content: '\e904';
  }
  .e-input-group-icon.e-input-right:before { /* csslint allow: adjoining-
classes */
    content: '\e913';
  }
  .e-input-group-icon.e-input-reload:before { /* csslint allow: adjoining-
classes */
    content: '\e837';
  }
  .e-input-group-icon.e-input-search:before { /* csslint allow: adjoining-
classes */
    content: '\e806';
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/floating-label-cs1" %}

Change the floating label color of the textbox in Vue Textbox component

You can change the floating label color of the TextBox for both success and warning validation states by applying the following CSS styles.

`css

/ *For Success state* /

.e-float-input.e-success label.e-float-text,

.e-float-input.e-success input:focus ~ label.e-float-text,

.e-float-input.e-success input:valid ~ label.e-float-text {

color: #22b24b;

}

/ *For Warning state* /

.e-float-input.e-warning label.e-float-text,

.e-float-input.e-warning input:focus ~ label.e-float-text,

.e-float-input.e-warning input:valid ~ label.e-float-text {

color: #ffca1c;

}

`

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
            <div class="e-input-group e-float-input e-success">
                <input type="text" required>
                <span class="e-float-line"></span>
                <label class="e-float-text">Success</label>
            </div>
            <div class="e-input-group e-float-input e-warning">
                <input type="text" required>
                <span class="e-float-line"></span>
                <label class="e-float-text">Warning</label>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
export default {
   data: function() {
        return { }
    },
```

```
    mounted: function() {
        // To get the all input fields and its container.
        let inputElement = document.querySelectorAll('.e-input-group .e-
input,.e-float-input.e-input-group input');
        // Add 'e-input-focus' class to the input for achive ripple effect
when focus on the input field.
        for (let i = 0; i < inputElement.length; i++) {
            inputElement[i].addEventListener("focus", function () {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.add('e-input-focus');
                } else {
                    this.parentNode.classList.add('e-input-focus');
                }
            });
            inputElement[i].addEventListener("blur", function () {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.remove('e-input-
focus');
                } else {
                    this.parentNode.classList.remove('e-input-focus');
                }
            });
        }
        // Add 'e-input-btn-ripple' class to the icon element for achive
ripple effect when click on the icon.
        var inputIcon = document.querySelectorAll('.e-input-group-icon');
        for (let i = 0; i < inputIcon.length; i++) {
            inputIcon[i].addEventListener('mousedown', function () {
                this.classList.add('e-input-btn-ripple');
            });
            inputIcon[i].addEventListener('mouseup', function () {
                let element = this;
                setTimeout(function () {
                    element.classList.remove('e-input-btn-ripple');
                }, 500);
            });
        }
    });
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  #input-container .e-input-group { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  #input-container .e-float-input { /* csslint allow: adjoining-classes */
    margin: 30px 0;
```

```
  }
  .e-float-input.e-success label.e-float-text{ /* csslint allow: adjoining-classes */
    color: #22b24b;
  }
  .e-float-input.e-success input:focus ~ label.e-float-text{ /* csslint allow: adjoining-classes */
    color: #22b24b;
  }
  .e-float-input.e-success input:valid ~ label.e-float-text { /* csslint allow: adjoining-classes */
    color: #22b24b;
  }
  .e-float-input.e-warning label.e-float-text{ /* csslint allow: adjoining-classes */
    color: #ffca1c;
  }
  .e-float-input.e-warning input:focus ~ label.e-float-text{ /* csslint allow: adjoining-classes */
    color: #ffca1c;
  }
  .e-float-input.e-warning input:valid ~ label.e-float-text { /* csslint allow: adjoining-classes */
    color: #ffca1c;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/validation-state-cs1" %}

Change the color of the textbox based on its value in Vue Textbox component

You can change the color of the TextBox by validating its value using regular expression in the keyup event for predicting the numeric values as demonstrated in the following code sample.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
            <label>Normal Input</label>
            <div class="e-input-group">
                <input class="e-input" type="text" placeholder="Enter numeric values" @keyup="onKeyup" />
            </div>
                <label> Floating Input </label>
            <div class="e-float-input">
                <input type="text" @keyup="onKeyup" required />
                <span class="e-float-line"> </span>
                <label class="e-float-text">Enter numeric values</label>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
export default {
   data: function() {
```

```
        return { }
    },
    methods: {
        // To Configure RegExp for predicting Numeric values on keyUp event
function
        onKeyup(e) {
            let str = e.target.value.match(/^[0-9]+$/);
            if (!((str && str.length > 0)) && e.target.value.length) {
            e.target.parentElement.classList.add('e-error');
            } else {
            e.target.parentElement.classList.remove('e-error');
            }
        }
    }
    mounted: function() {
        // To get the all input fields and its container.
        let inputElement = document.querySelectorAll('.e-input-group .e-
input,.e-float-input.e-input-group input');
        // Add 'e-input-focus' class to the input for achive ripple effect
when focus on the input field.
        for (let i = 0; i < inputElement.length; i++) {
            inputElement[i].addEventListener("focus", function () {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.add('e-input-focus');
                } else {
                    this.parentNode.classList.add('e-input-focus');
                }
            });
            inputElement[i].addEventListener("blur", function () {
                let parentElement = this.parentNode;
                if (parentElement.classList.contains('e-input-in-wrap')) {
                    parentElement.parentNode.classList.remove('e-input-
focus');
                } else {
                    this.parentNode.classList.remove('e-input-focus');
                }
            });
        }
    });
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
  .wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
  }
  #input-container .e-input-group { /* csslint allow: adjoining-classes */
    margin: 30px 0;
  }
  #input-container .e-float-input { /* csslint allow: adjoining-classes */
    margin: 30px 0;
```

```
  }
  .e-input-group.e-error .e-input { /* csslint allow: adjoining-classes */
    color: #f44336;
  }
  .e-float-input.e-error input { /* csslint allow: adjoining-classes */
    color: #f44336;
  }
  .wrap label { /* csslint allow: adjoining-classes */
    font-weight:bold;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/text-color-cs1" %}

Add floating label to read only textbox in Vue Textbox component

You can achieve floating label for read-only textboxes by adding/removing e-label-top and e-label-bottom classes to the label element

In this sample, click the update value button to fill the read-only textbox with value and float a label.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div class="row">
          <div class="e-float-input">
            <input class="e-input myField" id="myText" name="readonlyAttr"
type="text" readOnly>
            <span class="e-float-line"></span>
            <label class="e-float-text">Enter value</label>
          </div>
          <div class="row">
            <button class="e-btn update_value" id='valuebtn' v-
on:click="valuebtnClick">Set value</button>
            <button class="e-btn remove_value" id='removebtn' v-
on:click="removebtnClick">Remove value</button>
          </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
export default {
   data: function() {
        return { }
    },
    methods: {
        valuebtnClick: function(args){
            (document.getElementsByClassName('myField')[0]).value = '10';
            this.checkFloatingLabel('myText')
        },
        removebtnClick: function(args){
            (document.getElementsByClassName('myField')[0]).value = '';
            this.checkFloatingLabel('myText')
        },
        checkFloatingLabel: function(id) {
```

```
        var inputElement=  document.getElementById(id);
        var labelElement = inputElement.parentElement.querySelector('.e-
float-text');
        if (inputElement.value !== '') {
            labelElement.classList.remove('e-label-bottom');
            labelElement.classList.add('e-label-top');
        } else {
            labelElement.classList.remove('e-label-top');
            labelElement.classList.add('e-label-bottom');
        }
        var inputField = document.getElementById('myText');
        }
    },
    mounted: function() {
        // To get the all input fields and its container.
        let inputElement = document.querySelectorAll('.e-input-group .e-
input,.e-float-input.e-input-group input');
        // Add 'e-input-focus' class to the input for achive ripple effect
when focus on the input field.
        for (let i = 0; i < inputElement.length; i++) {
            inputElement[i].addEventListener("focus", function () {
                this.parentNode.classList.add('e-input-focus');
            });
            inputElement[i].addEventListener("blur", function () {
                this.parentNode.classList.remove('e-input-focus');
            });
        }
    }
};
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
.wrap {
  box-sizing: border-box;
  margin: 0 auto;
  padding: 20px 10px;
  width: 260px;
}
.update_value, .remove_value {
    margin-top: 20px;
}
.remove_value {
    margin-left: 10px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/read-only-cs1" %}

Customize the textbox background color and text color in Vue Textbox component

You can customize the textbox styles such as background-color, text-color and border-color by overriding its default styles.

To change the styles of the floating label, you must override the style to the input element.

APP.VUE

```
<template>
    <div class ='wrap'>
        <div id ='input-container'>
            <label>Normal Input</label>
            <div class="e-input-group">
                <input class="e-input" type="text" placeholder="First Name"
/>
            </div>
                <label> Floating Input </label>
            <div class="e-float-input">
                <input type="text" required />
                <span class="e-float-line"> </span>
                <label class="e-float-text">Last Name</label>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
export default {
   data: function() {
        return { }
    },
    mounted: function() {
        // To get the all input fields and its container.
        let inputElement = document.querySelectorAll('.e-input-group .e-
input,.e-float-input.e-input-group input');
        // Add 'e-input-focus' class to the input for achive ripple effect
when focus on the input field.
        for (let i = 0; i < inputElement.length; i++) {
            inputElement[i].addEventListener("focus", function () {
                this.parentNode.classList.add('e-input-focus');
            });
            inputElement[i].addEventListener("blur", function () {
                this.parentNode.classList.remove('e-input-focus');
            });
        }
    });
});
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
inputs/styles/material.css";
.wrap {
  box-sizing: border-box;
  margin: 0 auto;
  padding: 20px 10px;
  width: 260px;
}
.wrap label { /* csslint allow: adjoining-classes */
 font-weight:bold;
}
.wrap .e-float-input {/* csslint allow: adjoining-classes */
   margin:30px 0;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
.wrap .e-input-group {/* csslint allow: adjoining-classes */
   margin:25px 0;
}
/* To change the background-color and text-color for textbox */
.e-input-group,
.e-float-input,
.e-float-input.e-input-group { /* csslint allow: adjoining-classes */
  background : yellow;
  color: green;
}
/* To change the border-color of the textbox */
.e-input-group:not(.e-success):not(.e-warning):not(.e-error):not(.e-float-
icon-left),
.e-input-group:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-
float-icon-left) { /* csslint allow: adjoining-classes */
  border-color: #0800ff;
}
/* To change the border-color of the floating-label textbox */
.e-float-input input,
.e-float-input:hover:not(.e-input-group):not(.e-success):not(.e-
warning):not(.e-error):not(.e-disabled) input:not([disabled]) { /* csslint
allow: adjoining-classes */
    border-color: #0800ff;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/textbox/textbox-customize-cs1" %}

TimePicker

Getting Started with the Vue Timepicker Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Timepicker component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The list of dependencies required to use the Vue TimePicker component in your application is given below:

`javascript

|-- @syncfusion/ej2-vue-calendars

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-data

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-calendars

|-- @syncfusion/ej2-inputs

|-- @syncfusion/ej2-splitbuttons

Copyright © 2001 -2024 Syncfusion Inc.

|-- @syncfusion/ej2-lists

|-- @syncfusion/ej2-popups

|-- @syncfusion/ej2-buttons

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

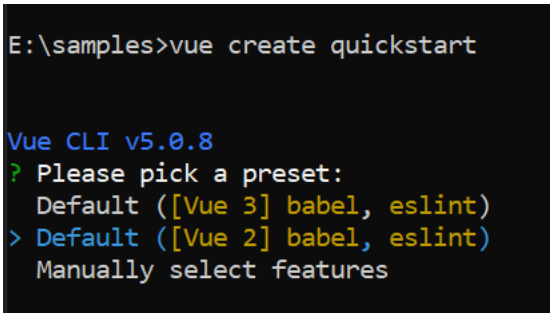
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Timepicker component as an example. Install the @syncfusion/ej2-vue-calendars package by running the following command:

Copyright © 2001 -2024 Syncfusion Inc.

`bash

npm install @syncfusion/ej2-vue-calendars --save

`

or

`bash

yarn add @syncfusion/ej2-vue-calendars

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Timepicker component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
</style>
```

Note: If you want to refer the combined component styles, please make use of our CRG (Custom Resource Generator) in your application.

Add Syncfusion Vue component

Follow the below steps to add the Vue Timepicker component using Composition API or Options API:

1\. First, import and register the Timepicker component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { TimePickerComponent as EjsTimepicker } from '@syncfusion/ej2-vue-
calendars';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { TimePickerComponent } from '@syncfusion/ej2-vue-calendars';
export default {
components: {
'ejs-timepicker': TimePickerComponent
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
}
</script>
```

2\. In the template section, define the Timepicker component with the placeholder property.

~/SRC/APP.VUE

```
<template>
<div id="app">
<div class='wrapper'>
<ejs-timepicker :placeholder="waterMark"></ejs-timepicker>
</div>
</div>
</template>
```

3\. Declare the value for the placeholder property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { TimePickerComponent as EjsTimepicker } from '@syncfusion/ej2-vue-
calendars';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data () {
return {
waterMark : 'Select a time'
}
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
      <div class='wrapper'>
        <ejs-timepicker :placeholder="waterMark"></ejs-timepicker>
      </div>
    </div>
</template>
<script setup>
import { TimePickerComponent as EjsTimepicker } from '@syncfusion/ej2-vue-
calendars';
const waterMark = 'Select a time';
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
  .wrapper {
    max-width: 250px;
    margin: 0 auto;
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
      <div class='wrapper'>
        <ejs-timepicker :placeholder="waterMark"></ejs-timepicker>
      </div>
    </div>
</template>
<script>
import { TimePickerComponent } from '@syncfusion/ej2-vue-calendars';
export default {
  components: {
    'ejs-timepicker': TimePickerComponent
  },
  data () {
    return {
      waterMark : 'Select a time'
    }
  }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
  .wrapper {
    max-width: 250px;
    margin: 0 auto;
  }
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/timepicker/getting-started-cs1" %}

Setting the value, min, and max time

The following example demonstrates how to set the value, min, and max time on initializing the TimePicker. The Vue TimePicker allows you to select the time value within a range from 7:00 AM to 4:00 PM.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
      <div class='wrapper'>
       <ejs-timepicker :min="minDate" :max="maxDate" :value="timeVal"></ejs-
timepicker>
      </div>
    </div>
</template>
<script setup>
import { TimePickerComponent as EjsTimepicker } from '@syncfusion/ej2-vue-
calendars';
const minDate =  new Date("05/07/2017 7:00 AM");
const maxDate = new Date("05/07/2017 4:00 PM");
const timeVal = new Date("05/27/2017 1:00 PM");

</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
   .wrapper {
    max-width: 250px;
    margin: 0 auto;
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
      <div class='wrapper'>
       <ejs-timepicker :min="minDate" :max="maxDate" :value="timeVal"></ejs-
timepicker>
      </div>
    </div>
</template>
<script>
import { TimePickerComponent } from '@syncfusion/ej2-vue-calendars';
export default {
  components: {
    'ejs-timepicker': TimePickerComponent
  },
  data () {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        return {
           minDate : new Date("05/07/2017 7:00 AM"),
           maxDate : new Date("05/07/2017 4:00 PM"),
           timeVal : new Date("05/27/2017 1:00 PM")
        }
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
   .wrapper {
    max-width: 250px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/min-max-cs1" %}

Setting the time format

Time formats is a way of representing the time value in different string format in textbox and popup list. By default, the TimePicker's format is based on the culture. You can also customize the format by using the format property. To know more about the time format standards, refer to the Date and Time Format section.

The following example demonstrates the TimePicker component in 24 hours format with 60 minutes interval. The time interval is set to 60 minutes by using the step property.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
    <div class='wrapper'>
    <ejs-timepicker :step="timeStep" :format="timeFormat"
:value="timeVal"></ejs-timepicker></div>
  </div>
</template>
<script setup>
import { TimePickerComponent as EjsTimepicker } from '@syncfusion/ej2-vue-
calendars';
const timeStep = 60;
const timeFormat = 'HH:mm';
const timeVal = new Date();
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
   .wrapper {
    max-width: 250px;
    margin: 0 auto;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
    <div id="app">
    <div class='wrapper'>
    <ejs-timepicker :step="timeStep" :format="timeFormat"
:value="timeVal"></ejs-timepicker></div>
  </div>
</template>
<script>
import { TimePickerComponent } from '@syncfusion/ej2-vue-calendars';
export default {
  components: {
    'ejs-timepicker': TimePickerComponent
  },
  data () {
        return {
           timeStep : 60,
           timeFormat : 'HH:mm',
           timeVal : new Date()
        }
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
   .wrapper {
    max-width: 250px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/format-cs1" %}

Once the time format property is defined, it will be applicable to all the cultures.

See Also

- Render TimePicker with min and max time
- How to achieve validation with TimePicker
- Render TimePicker with specific culture

Getting Started with the Vue TimePicker Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue TimePicker component using the Composition API / Options API.

Copyright © 2001 -2024 Syncfusion Inc.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

Copyright © 2001 -2024 Syncfusion Inc.

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue TimePicker component as an example. To use the Vue TimePicker component in the project, the @syncfusion/ej2-vue-calendars package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-calendars --save

`

or

`bash

yarn add @syncfusion/ej2-vue-calendars

`

Copyright © 2001 -2024 Syncfusion Inc.

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the TimePicker component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue TimePicker component using Composition API or Options API:

1.First, import and register the TimePicker component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { TimePickerComponent as EjsTimepicker } from "@syncfusion/ej2-vue-
calendars";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { TimePickerComponent } from "@syncfusion/ej2-vue-calendars";
//Component registeration
export default {
name: "App",
components: {
'ejs-timepicker' : TimePickerComponent,
}
}
</script>
```

2.In the template section, define the TimePicker component with the dataSource property and column definitions.

~/SRC/APP.VUE

```
<template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<div class="control_wrapper">
<ejs-timepicker></ejs-timepicker>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-timepicker></ejs-timepicker>
</div>
</template>
<script setup>
import { TimePickerComponent as EjsTimepicker } from "@syncfusion/ej2-vue-
calendars";
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
.control_wrapper {
max-width: 250px;
margin: 0 auto;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-timepicker></ejs-timepicker>
</div>
</template>
<script>
import { TimePickerComponent } from "@syncfusion/ej2-vue-calendars";
//Component registeration
export default {
name: 'App',
components: {
"ejs-timepicker": TimePickerComponent
},
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
.control_wrapper {
max-width: 250px;
margin: 0 auto;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

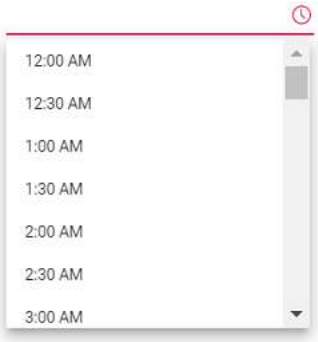
or

`bash

yarn run dev

`

The output will appear as follows:



Setting the value, min, and max time

The following example demonstrates how to set the value, min, and max time on initializing the TimePicker. The TimePicker allows you to select the time value within a range from 7:00 AM to 4:00 PM.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrapper'>
<ejs-timepicker :min="data[0].minDate" :max="data[0].maxDate"
:value="data[0].timeVal"></ejs-timepicker>
</div>
</div>
</template>
<script setup>
import { TimePickerComponent as EjsTimepicker } from "@syncfusion/ej2-vue-
calendars";
const data = [{ minDate : new Date("05/07/2017 7:00 AM"),
maxDate : new Date("05/07/2017 4:00 PM"),
timeVal : new Date("05/27/2017 1:00 PM") }]
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

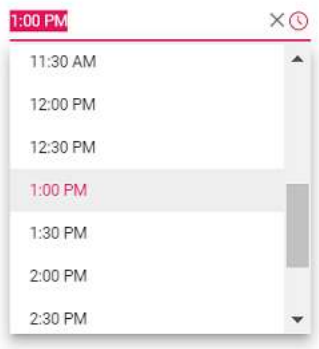
```
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
.wrapper {
max-width: 250px;
margin: 0 auto;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrapper'>
<ejs-timepicker :min="minDate" :max="maxDate" :value="timeVal"></ejs-
timepicker>
</div>
</div>
</template>
<script>
import { TimePickerComponent } from "@syncfusion/ej2-vue-calendars";
//Component registeration
export default {
name: 'App',
components: {
"ejs-timepicker": TimePickerComponent
},
data () {
return {
minDate : new Date("05/07/2017 7:00 AM"),
maxDate : new Date("05/07/2017 4:00 PM"),
timeVal : new Date("05/27/2017 1:00 PM")
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
.wrapper {
max-width: 250px;
margin: 0 auto;
}
</style>
```

The output will appear as follows:

Copyright © 2001 -2024 Syncfusion Inc.



Setting the time format

Time formats is a way of representing the time value in different string format in textbox and popup list. By default, the TimePicker's format is based on the culture. You can also customize the format by using the format property. To know more about the time format standards, refer to the Date and Time Format section.

The following example demonstrates the TimePicker component in 24 hours format with 60 minutes interval. The time interval is set to 60 minutes by using the step property.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrapper'>
<ejs-timepicker :step="data[0].timeStep" :format="data[0].timeFormat"
:value="data[0].timeVal"></ejs-timepicker>
</div>
</div>
</template>
<script setup>
import { TimePickerComponent as EjsTimepicker } from "@syncfusion/ej2-vue-
calendars";
const data = [{ timeStep : 60,
timeFormat : 'HH:mm',
timeVal : new Date() }]
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
.wrapper {
max-width: 250px;
margin: 0 auto;
}
</style>
```

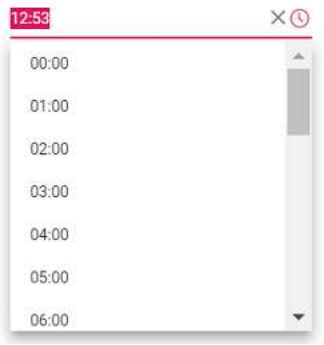
OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class='wrapper'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<ejs-timepicker :step="timeStep" :format="timeFormat"
:value="timeVal"></ejs-timepicker>
</div>
</div>
</template>
<script>
import { TimePickerComponent } from "@syncfusion/ej2-vue-calendars";
//Component registeration
export default {
name: 'App',
components: {
"ejs-timepicker": TimePickerComponent
},
data () {
return {
timeStep : 60,
timeFormat : 'HH:mm',
timeVal : new Date()
}
}
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
.wrapper {
max-width: 250px;
margin: 0 auto;
}
</style>
```

Output be like the below.



Once the time format property is defined, it will be applicable to all the cultures.

See Also

- Render TimePicker with min and max time
- How to achieve validation with TimePicker
- Render TimePicker with specific culture

Copyright © 2001 -2024 Syncfusion Inc.

Time range in Vue Timepicker component

TimePicker provides an option to select a time value within a specified range by using the min and max properties. The min value should always be lesser than the max value.

When the min and max properties are configured and the selected time value is out-of-range or invalid, then the model value will be set to out of range time value or null respectively with highlighted error class to indicates the time is out of range or invalid.

The value property depends on the min/max with respect to strictMode property.

The following example allows you to select a time value within a range of 9:00 AM to 11:30 AM.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-timepicker id='timepicker' :value='timeval' :min='minTime'
:max='maxTime'></ejs-timepicker>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { TimePickerPlugin } from "@syncfusion/ej2-vue-calendars";
Vue.use(TimePickerPlugin);
export default {
  data () {
    return{
        timeval:new Date('3/8/2017 11:00 AM'),
        minTime: new Date('3/8/2017 9:00 AM'),
        maxTime: new Date('3/8/2017 11:30 AM')
    }
  }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
 .wrap {
    margin: 35px auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/range-cs1" %}

If the value of min or max property is changed through code behind you have to update the value property to set within the range.

Copyright © 2001 -2024 Syncfusion Inc.

Time masking in Vue Timepicker component

TimePicker has enableMask property that provides the option to enable the built-in date masking support. Also, you must inject the MaskedDateTime module to enable the masking support.

APP.VUE

```
<template>
<div id="app">
        <div class='wrapper'>
        <ejs-timepicker id="timepicker" :enableMask="true"></ejs-timepicker>
     </div>
  </div>
</template>
<script>
import Vue from "vue";
import { TimePickerPlugin, TimePicker, MaskedDateTime } from
"@syncfusion/ej2-vue-calendars";
TimePicker.Inject(MaskedDateTime)
Vue.use(TimePickerPlugin);
export default Vue.extend({
  data: function() {
    return {
    };
  },
  provide: {
      timepicker: [MaskedDateTime]
  }
});
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
  .wrapper {
    max-width: 250px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/mask-module-cs1" %}

The mask pattern is defined based on the provided time format to the component. If the format is not specified, the mask pattern is formed based on the default format of the current culture.

| **Keys** | **Actions** |

| --- | --- |

| Up / Down arrows | To increment and decrement the selected portion of the time. |

| Left / Right arrows and Tab | To navigate the selection from one portion to next portion |

The following example demonstrates default and custom format of TimePicker component with mask.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<div id="app">
        <div class='wrapper1'>
         <!-- Specifies the masked timepicker without format property. -->
         <ejs-timepicker id="timepicker" :enableMask="true" ></ejs-
timepicker>
     </div>
     <div class='wrapper2'>
          <!-- Specifies the masked timepicker with format property. -->
         <ejs-timepicker id="format" :enableMask="true"
:format="dateFormat"></ejs-timepicker>
     </div>
  </div>
</template>
<script>
import Vue from "vue";
import { TimePickerPlugin, TimePicker, MaskedDateTime } from
"@syncfusion/ej2-vue-calendars";
TimePicker.Inject(MaskedDateTime)
Vue.use(TimePickerPlugin);
export default{
  data() {
    return {
      dateFormat: 'hh:mm a'
    };
  },
  provide: {
      timepicker: [MaskedDateTime]
  }
};
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
#wrapper1{
  min-width: 250px;
    float: left;
    margin-left: 100px;
}
#wrapper2{
  min-width: 250px;
    float: right;
     margin-right: 100px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/mask-module-cs2" %}

Configure Mask Placeholder

You can change mask placeholder value through property `maskPlaceholder`. By default , it takes the full name of time co-ordinates such as `hour`, `minute` and `second`.

Copyright © 2001 -2024 Syncfusion Inc.

While changing to a culture other than English, ensure that locale text for the concerned culture is loaded through load method of L10n class for mask placeholder values like below.

`ts

//Load the L10n from ej2-base

import { L10n } from '@syncfusion/ej2-base';

//load the locale object to set the localized mask placeholder value

L10n.load({

'de': {

'timepicker': { hour: 'Stunde' ,minute: 'Minute', second:'Sekunde' }

}

});

`

The following example demonstrates default and customized mask placeholder value.

APP.VUE

```
<template>
<div id="app">
        <div class='wrapper1'>
        <!-- Specifies the masked timepicker without mask placeholder. -->
        <ejs-timepicker id="timepicker" :enableMask="true" ></ejs-
timepicker>
     </div>
     <div class='wrapper2'>
        <!-- Specifies the masked timepicker with mask placeholder. -->
        <ejs-timepicker id="placeholder" :enableMask="true"
:maskPlaceholder="maskPlaceholderValue"></ejs-timepicker>
     </div>
  </div>
</template>
<script>
import Vue from "vue";
import { TimePickerPlugin, TimePicker, MaskedDateTime } from
"@syncfusion/ej2-vue-calendars";
TimePicker.Inject(MaskedDateTime)
Vue.use(TimePickerPlugin);
export default{
  data() {
    return {
      maskPlaceholderValue: { hour: 'h', minute: 'm', second: 's' }
    };
  },
  provide: {
      timepicker: [MaskedDateTime]
  }
};
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
#wrapper1{
  min-width: 250px;
    float: left;
    margin-left: 100px;
}
#wrapper2{
  min-width: 250px;
  float: right;
  margin-right: 100px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/mask-module-cs3" %}

Globalization in Vue Timepicker component

Globalization is the combination of internalization and localization. You can adapt the component to various languages by parsing and formatting the date or number internationalization, and also add culture specific customization and translation to the text localization.

By default, the time format and meridian names are specific to the American English culture. It utilizes the Essential JavaScript 2 Internationalization package to parse and format the date object based on the culture by using the official UNICODE CLDR JSON data. It provides the loadCldr method to load culture specific CLDR JSON data. To use a different culture other than English, follow the steps below:

- Install the CLDR-Data package by using the following command (installs all the CLDR JSON data). To know more about CLDR-Data refer to the CLDR-Data link.

`

npm install cldr-data --save

`

Once the package is installed, you can find the culture specific JSON data under the location /node_modules/cldr-data.

- Import the installed CLDR JSON data into the app.vue file.
- Use the loadCldr

method to load the culture specific CLDR JSON data from the installed location to app.vue file.

- TimePicker displayed Sunday as the first day of week based on default culture ("en-US"). If you want to display the TimePicker with loaded culture's first day of week, you need to import weekdata.json file from the cldr-data/suppemental as given in the code example.

`ts

//Load the loadCldr from ej2-base

Copyright © 2001 -2024 Syncfusion Inc.

import { loadCldr } from '@syncfusion/ej2-base';

import * as numberingSystems from 'cldr-data/supplemental/numberingSystems.json';

import * as gregorian from 'cldr-data/main/de/ca-gregorian.json';

import * as numbers from 'cldr-data/main/de/numbers.json';

import * as weekData from 'cldr-data/supplemental/weekdata.json';// To load the culture based first day of week

loadCldr(numberingSystems, gregorian, numbers, weekData);

`

- Before changing to a culture other than English, ensure that the\ locale text for the concerned culture is loaded through L10n.loadmethod.

`ts

//Load the L10n from ej2-base

import { L10n } from '@syncfusion/ej2-base';

//load the locale object to set the localized placeholder value

L10n.load({

'de': {

'timepicker': { placeholder: 'Wählen Sie Zeit' }

}

});

`

- Set the culture by using the locale property.

The following example demonstrates the TimePicker in German culture.

APP.VUE

```
<template>
    <div id="app">
      <div class='wrapper'>
        <ejs-timepicker id="timepicker"  locale="de" ></ejs-timepicker>
      </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { loadCldr,L10n} from '@syncfusion/ej2-base';
import { TimePickerPlugin } from '@syncfusion/ej2-vue-calendars';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
Vue.use(TimePickerPlugin);
loadCldr(numberingSystems, gregorian, numbers);
L10n.load({
    'de': {
        'timepicker': { placeholder: 'Wählen Sie Zeit' }
    }
});
export default {}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
 .wrapper {
    max-width: 250px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/locale-cs1" %}

Right-To-Left

The TimePicker supports RTL (right-to-left) functionality for languages like Arabic and Hebrew to displays the text in the right-to-left direction. Use enableRtl property to set the RTL direction.

The code example demonstrates the TimePicker component in Arabic culture. It also explains how to set localized text to the placeholder using L10n.load method.

`

APP.VUE

```
<template>
    <div id="app">
      <div class='wrapper'>
        <ejs-timepicker id="timepicker"  :locale="locale" :enableRtl="rtl"
></ejs-timepicker>
      </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { loadCldr,L10n} from '@syncfusion/ej2-base';
import { TimePickerPlugin } from '@syncfusion/ej2-vue-calendars';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(TimePickerPlugin);
loadCldr(numberingSystems, gregorian, numbers);
L10n.load({
    'ar': {
        'timepicker': {  placeholder: 'حدد الوقت' }
    }
});
export default {
    data () {
       return {
           locale: "ar",
           rtl: true
        }
    }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
 .wrapper {
    max-width: 250px;
    margin: 0 auto;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/locale-rtl-cs1" %}

Strict mode in Vue Timepicker component

Enable Strict Mode

The strictMode is an act that allows you to enter only valid time value within the specified min/max range in the textbox. If the time value is invalid, the component value sets to the previous value. If the time value is out of range, the component sets the time value to min/max value.

The following example demonstrates the TimePicker in strictMode with min/max range of 10:00 AM to 4:00 PM . It allows you to enter only valid time within the specified range. If you enter the out-of-range value like 8:00 PM, the value sets to the max time 4:00 PM as the value 8:00 PM is greater than max value of 4:00 PM. If you enter invalid time value like 9:00 tt, the value sets to the previous value.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-timepicker id='timepicker' :strictMode='Strict'
:value='timeval' :min='minTime' :max='maxTime'></ejs-timepicker>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { TimePickerPlugin } from "@syncfusion/ej2-vue-calendars";
Vue.use(TimePickerPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data () {
    return{
        Strict: true,
        timeval:new Date('7/18/2017 3:00 PM'),
        minTime: new Date('7/18/2017 10:00 AM'),
        maxTime: new Date('7/18/2017 4:00 PM')
    }
  }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
 .wrap {
    margin: 35px auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/strict-cs2" %}

Disable Strict Mode

By default, the TimePicker act in strictMode `false` state, that allows to enter the invalid or out-of-range time in textbox.

If the time is out-of-range or invalid, then the model value will be set to `out of range` time value or `null` respectively with highlighted `error` class to indicates the time is out of range or invalid.

The following example demonstrates the `strictMode` as `false`. Here, it allows to enter the valid or invalid value in textbox. If you are entering the out-of-range or invalid time value, then the model value will be set to`out of range` time value or `null` respectively with highlighted `error` class to indicates the time is out of range or invalid.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-timepicker id='timepicker' :value='timeval' :min='minTime'
:max='maxTime'></ejs-timepicker>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { TimePickerPlugin } from "@syncfusion/ej2-vue-calendars";
Vue.use(TimePickerPlugin);
export default {
  data () {
    return{
        timeval:new Date('7/18/2017 3:00 PM'),
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        minTime: new Date('7/18/2017 10:00 AM'),
        maxTime: new Date('7/18/2017 4:00 PM')
    }
  }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import "../node_modules/@syncfusion/ej2-vue-calendars/styles/material.css";
 .wrap {
    margin: 35px auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/strict-cs3" %}

If the value of min or max property is changed through code behind, update the value property to set within the range.

Accessibility in Vue Timepicker component

The TimePicker component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the TimePicker component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The web accessibility makes web applications and its content more accessible to people with disabilities without any barriers. It especially it tracks the dynamic value changes and DOM changes.

The TimePicker component has covered the WAI-ARIA specifications with the following list of WAI-ARIA attributes: aria-haspopup, aria-selected, aria-disabled, aria-activedescendant, aria-expanded, aria-owns, and aria-autocomplete.

Here in TimePicker, the combobox plays the role of input element, and the listbox plays the role of popup element.

- **Aria-haspopup**: Provides the information about whether this element display a pop-up window or not.
- **Aria-selected**: Indicates the current selected value of the TimePicker component.
- **Aria-disabled**: Indicates disabled state of the TimePicker component.
- **Aria-expanded**: Indicates the expanded state of the popup.
- **Aria-autocomplete**: Indicates whether user input completion suggestions are provided or not.
- **Aria-owns**: Attribute that creates a parent/child relationship between two DOM element in the accessibility layer.
- **Aria-activedescendent**: Attribute that helps in managing the current active child of the TimePicker component.
- **Role**: Attribute that gives assistive technology information for handling each element in a widget.

Keyboard Interaction

Keyboard accessibility is one of the most important aspects of web accessibility. Disabled people like blind and those who have motor disabilities or birth defects use keyboard shortcuts more than the mouse.

Copyright © 2001 -2024 Syncfusion Inc.

The TimePicker component has built-in keyboard accessibility support by following the WAI-ARIA practices.

It supports the following list of shortcut keys to interact with the TimePicker control.

| **Press** | **To do this** |

| --- | --- |

| Upper Arrow | Navigate and select the previous item. |

| Down Arrow | Navigate and select the next item. |

| Left Arrow | Move the cursor towards arrow key pressed direction. |

| Right Arrow | Move the cursor towards arrow key pressed direction. |

| Home | Navigate and select the first item. |

| End | Navigate and select the last item. |

| Enter | Select the currently focused item and close the popup. |

| Alt + Upper Arrow | Close the popup. |

| Alt + Down Arrow | Open the popup. |

| Esc | Close the popup. |

In the below sample use the alt+t keys to focus the TimePicker component.

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-timepicker id='timepicker' ref="time"></ejs-timepicker>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { TimePickerPlugin } from "@syncfusion/ej2-vue-calendars";
Vue.use(TimePickerPlugin);
export default {
  mounted () {
      let TimeObj = this.$refs.time;
       document.onkeyup = function (e) {
            if (e.altKey && e.keyCode === 84) {
            // press alt+t to focus the control.
            TimeObj.$el.focus();
            }
        }
    }
}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
  @import "../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
 .wrap {
    margin: 35px auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/access-cs1" %}

Ensuring accessibility

The TimePicker component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the TimePicker component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the TimePicker component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/time-picker.html" %}

See also

- Accessibility in Syncfusion Vue components

Two way binding in Vue Timepicker component

Two-way binding can be achieved by using the v-model directive in Vue. In the following sample, when you change the value in one TimePicker component, v-model will automatically update the value in the other TimePicker.

The following example demonstrates how to set the two-way-binding in the TimePicker.

APP.VUE

```
<template>
<div id="app">
    <div id="wrapper1">
        <ejs-timepicker id="timepicker" v-model="date"></ejs-timepicker>
    </div>
    <div id="wrapper2">
        <ejs-timepicker id="timepicker1" v-model="date"></ejs-timepicker>
    </div>
</div>
</template>
<script>
import Vue from "vue";
import { TimePickerPlugin } from "@syncfusion/ej2-vue-calendars";
Vue.use(TimePickerPlugin);
export default {
  data: function() {
    return {
      date: new Date()
    };
  }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
#wrapper1{
    min-width: 250px;
    float: left;
    margin-left: 100px;
}
#wrapper2{
    min-width: 250px;
    float: right;
     margin-right: 100px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/two-way-cs1" %}

Style appearance in Vue Timepicker component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of TimePicker wrapper element

Use the following CSS to customize the appearance of wrapper element.

`

/ To specify height and font size /

.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input, .e-input-group textarea.e-input, .e-input-group.e-control-wrapper textarea.e-input {

font-size: 20px;

height: 40px;

}

`

Customizing the TimePicker icon element

Use the following CSS to customize the TimePicker icon element

`

/ To specify background color and font size /

.e-time-wrapper .e-time-icon.e-icons, *.e-control-wrapper.e-time-wrapper .e-time-icon.e-icons {

font-size: 20px;

background-color: beige;

}

`

Copyright © 2001 -2024 Syncfusion Inc.

Customizing the TimePicker popup

Use the following CSS to customize the TimePicker popup

```
/ To specify height /
.e-timepicker.e-popup {
height: 100px;
}
```

Customizing the TimePicker popup content

Use the following CSS to customize the TimePicker popup content

```
/ To specify height /
.e-timepicker.e-popup .e-list-parent.e-ul li.e-list-item {
background-color: beige;
font-size: 20px;
}
```

Full screen mode support in mobiles and tablets

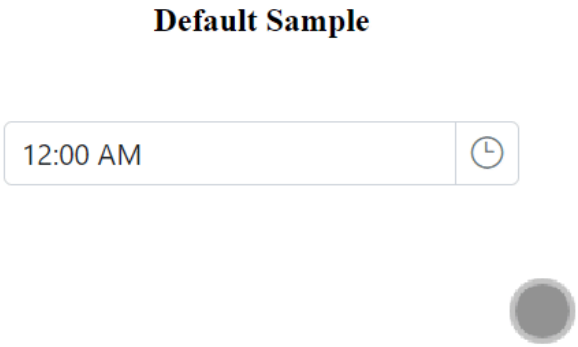
The DateTimePicker component's full-screen mode feature enables users to view the component popup element in full-screen mode on mobile devices with improved visibility and a better user experience. It is important to mention that this feature is exclusively available for mobile and tablet devices in both landscape and portrait orientations. To activate the full screen mode within the TimePicker component, simply set the fullScreenMode API value to true. This action will extend the popup element to occupy the entire screen on mobile devices.

```html
<template>
<div id="app">
<ejs-timepicker :fullScreenMode="mobileMode" ></ejs-timepicker>
</div>
</template>
<script>
import Vue from 'vue';
import { TimePickerPlugin } from '@syncfusion/ej2-vue-calendars';
Vue.use(TimePickerPlugin);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
data () {
return {
mobileMode : true
}
}
}
</script>
`
```

Copyright © 2001 -2024 Syncfusion Inc.



Copyright © 2001 -2024 Syncfusion Inc.

How To

Css customization in Vue Timepicker component

TimePicker allows you to customize the textbox and popup list appearance to suit your application by using cssClass property.

The below sample demonstrates customization of text appearance in a textbox, popup button, and popup list along with hover and active

state by using e-custom-style class. Following is the list of available classes used to customize the entire TimePicker component.

| **Class Name** | **Description** |

| --- | --- |

| e-time-wrapper | Applied to TimePicker wrapper element. |

| e-timepicker | Applied to input element and TimePicker popup element. |

| e-time-wrapper.e-timepicker | Applied to input element only. |

| e-input-group-icon.e-time-icon | Applied to popup button. |

| e-float-text | Applied to floating label text element. |

| e-popup | Applied to popup element. |

| e-timepicker.e-popup | Applied to TimePicker popup element only. |

| e-list-parent | Applied to popup UL element. |

| e-timepicker.e-list-parent | Applied to TimePicker popup UL element only. |

| e-list-item | Applied to LI elements. |

| e-hover | Applied to LI element hovering time. |

| e-active | Applied to active LI element. |

APP.VUE

```
<template>
  <div id="app">
        <div class='wrap'>
           <ejs-timepicker id='timepicker' placeholder='Select a time'
cssClass='e-custom-style'></ejs-timepicker>
        </div>
  </div>
</template>
<script>
import Vue from "vue";
import { TimePickerPlugin } from "@syncfusion/ej2-vue-calendars";
Vue.use(TimePickerPlugin);
export default {}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
 .wrap {
    margin: 35px auto;
    width: 240px;
}
/*customize the input element text color*/
.e-time-wrapper.e-custom-style #element { /* csslint allow: adjoining-
classes */
    display: block;
    color: blue;
}
/*customize the floating label and popup button text color*/
.e-time-wrapper.e-custom-style .e-float-text.e-label-bottom, /* csslint
allow: adjoining-classes */
.e-time-wrapper.e-custom-style .e-input-group-icon.e-time-icon.e-icons { /*
csslint allow: adjoining-classes */
    color: blue;
}
/*customize the input element text selection*/
.e-time-wrapper.e-custom-style.e-input-group::before, /* csslint allow:
adjoining-classes */
.e-time-wrapper.e-custom-style.e-input-group::after, /* csslint allow:
adjoining-classes */
.e-time-wrapper.e-custom-style.e-input-group .e-timepicker::selection { /*
csslint allow: adjoining-classes */
    background: blue;
}
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul, /* csslint allow:
adjoining-classes */
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul .e-list-item { /*
csslint allow: adjoining-classes */
    background-color: #c0ebff;
}
/*customize the list item hover color*/
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul .e-list-item.e-
hover, /* csslint allow: adjoining-classes */
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul .e-list-item.e-
active.e-hover { /* csslint allow: adjoining-classes */
    background-color: blue;
    color: #fff;
}
/*customize the active element text color*/
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul .e-list-item.e-
active { /* csslint allow: adjoining-classes */
    color:#333;
    background-color: #fff;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/strict-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Client side validation using form validator in Vue Timepicker component

To achieve client side validation in a TimePicker component, use Essential JavaScript 2 FormValidator. It provides an option to customize feedback error messages to the corresponding fields for taking action and resolving the issue.

In the following example, the required field validation is implemented by mapping the name attribute value to the rules property. It validates the TimePicker component and displays the validation message when the textbox value is empty during form post back or focus out.

APP.VUE

```
<template>
  <div id="app">
    <div class='wrap'>
      <form id="form-element" class="form-horizontal">
           <div class="form-group" style="padding-top: 11px;">
                <div class="col-lg-8">
                     <ejs-timepicker id="time" name="time"
:change="onTimeChange" class="form-control" placeholder='Select a
Time'></ejs-timepicker>
                </div>
            </div>
      </form>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { TimePickerPlugin } from "@syncfusion/ej2-vue-calendars";
import { FormValidator } from '@syncfusion/ej2-inputs';
Vue.use(TimePickerPlugin);
export default {
    data: function() {
        return {
          formObj: '',
          options : {
            customPlacement: function(inputelement, errorElement) {
                var parentElement = inputelement.closest('.form-group');
                parentElement.appendChild(errorElement);
            },
            rules: {
                'time': {
                    required: true
                }
            }
          }
        }
    },
  mounted: function () {
      this.formObj = new FormValidator('#form-element', this.options);
  },
  methods: {
      onTimeChange: function() {
          this.formObj.validate();
      }
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
<style>
  @import '../node_modules/@syncfusion/ej2-base/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
  @import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
  @import "../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
 .wrap {
    margin: 0 auto;
    width: 240px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/timepicker/validation-cs1" %}

Toast

Getting Started with the Vue Toast Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Toast component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of dependencies are required to use the toast component in your application:

`js

|-- @syncfusion/ej2-vue-notifications

|-- @syncfusion/ej2-popups

|-- @syncfusion/ej2-buttons

|-- @syncfusion/ej2-vue-base

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

Copyright © 2001 -2024 Syncfusion Inc.

`bash

yarn global add @vue/cli

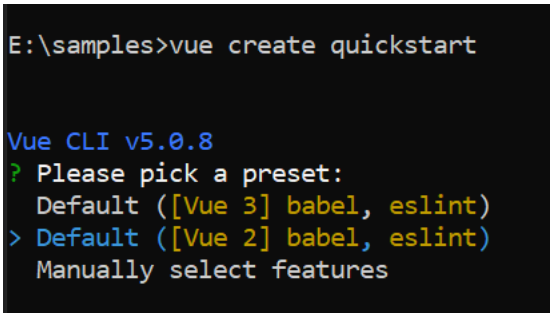
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Toast component as an example. Install the @syncfusion/ej2-vue-notifications package by running the following command:

`bash

npm install @syncfusion/ej2-vue-notifications --save

`

or

`bash

yarn add @syncfusion/ej2-vue-notifications

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Toast component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Toast component using Composition API or Options API:

1\. First, import and register the Toast component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { ToastComponent as EjsToast } from '@syncfusion/ej2-vue-
notifications';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { ToastComponent } from '@syncfusion/ej2-vue-notifications';
export default {
components: {
'ejs-toast': ToastComponent
}
}
</script>
```

2\. In the template section, define the Toast component with content and title property.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-toast ref='defaultRef' title='Matt sent you a friend request' timeOut=0
content='Hey, wanna dress up as wizards and ride our hoverboards?'></ejs-
toast>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
       <ejs-toast ref='defaultRef' title='Matt sent you a friend request'
timeOut=0 content='Hey, wanna dress up as wizards and ride our
hoverboards?'></ejs-toast>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script setup>
import { ToastComponent as EjsToast } from '@syncfusion/ej2-vue-
notifications';
import { ref, onMounted } from 'vue';
const defaultRef = ref(null);
onMounted (() => {
      defaultRef.value.show();
  });
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
       <ejs-toast ref='defaultRef' title='Matt sent you a friend request'
timeOut=0 content='Hey, wanna dress up as wizards and ride our
hoverboards?'></ejs-toast>
  </div>
</template>
<script>
import { ToastComponent } from '@syncfusion/ej2-vue-notifications';
export default {
  components: {
    'ejs-toast': ToastComponent
  },
  name: 'app',
  mounted: function() {
      this.$refs.defaultRef.show();
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

Copyright © 2001 -2024 Syncfusion Inc.

`

{% previewsample "page.domainurl/code-snippet/toast/getting-started-cs1" %}

See Also

- Render different types of toast

+N> You can refer to our Vue Toast feature tour page for its groundbreaking feature representations. You can also explore our Vue Toast Example that shows you how to render the Toast in Vue.

Getting Started with the Vue Toast Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Toast component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

Copyright © 2001 -2024 Syncfusion Inc.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

Copyright © 2001 -2024 Syncfusion Inc.

This article uses the Vue Toast component as an example. To use the Vue Toast component in the project, the @syncfusion/ej2-vue-notifications package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-notifications --save

`

or

`bash

yarn add @syncfusion/ej2-vue-notifications

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Toast component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue Toast component using Composition API or Options API:

1.First, import and register the Toast component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { ToastComponent as EjsToast } from "@syncfusion/ej2-vue-
notifications";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { ToastComponent } from "@syncfusion/ej2-vue-notifications";
//Component registeration
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
name: "App",
components: {
"ejs-toast": ToastComponent,
}
}
</script>
```

2.In the template section, define the Toast component with the dataSource property and column definitions.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-toast ref='defaultRef' title='Matt sent you a friend request' timeOut=0
content='Hey, wanna dress up as wizards and ride our hoverboards?'></ejs-
toast>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-toast
id="toast_default"
ref="defaultRef"
title="Adaptive Tiles Meeting"
timeOut="0"
icon="e-meeting"
content="Conference Room 01 / Building 135 10:00 AM-10:30 AM"
></ejs-toast>
</div>
</template>
<script setup>
import { ToastComponent as EjsToast } from "@syncfusion/ej2-vue-
notifications";
import { ref, onMounted } from 'vue';
const defaultRef = ref(null);
onMounted(() => {
defaultRef.value.show();
});
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
#toast_default .e-meeting::before {
content: "\e705";
font-size: 17px;
}
.bootstrap4 #toast_default .e-meeting::before {
content: "\e763";
font-size: 20px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-toast
id="toast_default"
ref="defaultRef"
title="Adaptive Tiles Meeting"
timeOut="0"
icon="e-meeting"
content="Conference Room 01 / Building 135 10:00 AM-10:30 AM"
></ejs-toast>
</div>
</template>
<script>
import { ToastComponent } from "@syncfusion/ej2-vue-notifications";
export default {
name: "App",
components: {
"ejs-toast": ToastComponent,
},
mounted: function () {
this.$refs.defaultRef.show();
},
data: function () {
return {};
},
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
#toast_default .e-meeting::before {
content: "\e705";
font-size: 17px;
}
.bootstrap4 #toast_default .e-meeting::before {
content: "\e763";
font-size: 20px;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

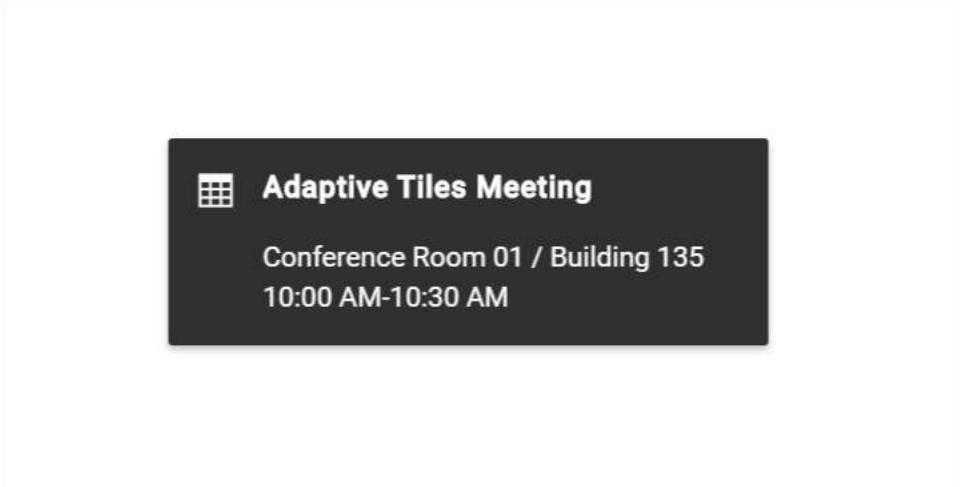
`bash

yarn run dev

`

Web server will be initiated, Open the quick start app in the browser at port localhost:8080.

The output will appear as follows:



Config in Vue Toast component

This section explains the steps required to customize the appearance of the toast using built-in APIs.

Title and content template

Toast can be created with the notification message. The message contains title and content of the toasts. The title and contents are adaptable in any resolution.

The Title or Content property can be given as HTML element/element ID to a string that can be displayed as a toast.

APP.VUE

```
<template>
  <div id="app">
       <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
       <ejs-toast ref='defaultRef' title='Matt sent you a friend request'
content='You have a new friend request yet to accept'
:position='position'></ejs-toast>
  </div>
</template>
<script>
import Vue from 'vue';
import { ToastPlugin } from '@syncfusion/ej2-vue-notifications';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
    return {
```

```
        position: { X: 'Center' }
    }
  },
  mounted: function() {
      this.$refs.defaultRef.show();
  },
  methods: {
      showBtnClick: function(args){
          this.$refs.defaultRef.show();
      }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/config/title-content-cs1" %}

Specifying custom target

By default, the toast can be rendered in the document body. You can change the target position for toast rendering using the target property. Based on the target, the position will be updated.

Close button

By default, the showCloseButton is not enabled. You can enable it by setting the true value. Before expiring the toast, you can use this button to close or destroy toasts manually.

Progress bar

By default, the showProgressBar is not enabled. If it is enabled, it can visually indicate how long to get toast expires. Based on the timeOut property, progress bar will appear.

Progress bar direction

By default, the progressDirection is set to "Rtl" and it will appear from right to left direction. You can change the progressDirection to "Ltr" to make it appear from left to right direction.

Newest on top

By default, the newly created toasts will append next with existing toasts. You can change the sequence like inserting before the toast by enabling the newestOnTop.

Here, The following sample demonstrates the combination of the target, showCloseButton, showProgressBar, and newestOnTop properties in toast.

APP.VUE

```
<template>
  <div id="app">
       <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
       <ejs-toast ref='defaultRef' title='File Downloading' content='<div
class="progress"><span style="width: 80%"></span></div>'
:position='position' showCloseButton=true target='#toast_target'
newestOnTop=true showProgressBar=true progressDirection="Ltr"></ejs-toast>
       <br/><br/>
       <div id='toast_target'></div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  </div>
</template>
<script>
import Vue from 'vue';
import { ToastPlugin } from '@syncfusion/ej2-vue-notifications';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
    return {
        position: { X: 'Center' }
    }
  },
  mounted: function() {
      this.$refs.defaultRef.show();
  },
  methods: {
      showBtnClick: function(args){
          this.$refs.defaultRef.show();
      }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
<style lang="scss">
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-toast-message {
    width: 100%;
}
.progress {
    height: 20px;
    position: relative;
    margin: 20px 0 20px 0;
    background: #555;
    -moz-border-radius: 25px;
    -webkit-border-radius: 25px;
    border-radius: 25px;
    box-shadow: inset 0 -1px 1px rgba(255, 255, 255, 0.3);
}
.progress span {
    background-color: #f0a3a3;
    background-image: -webkit-gradient(linear, left top, left bottom, color-
stop(0, #f0a3a3), color-stop(1, #f42323));
    display: block;
    height: 100%;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    border-radius: 10px;
    width: 50%;
    position: relative;
    overflow: hidden;
}
.progress span::after {
  background-image: -webkit-gradient(linear, 0 0, 100% 100%,
    color-stop(.25, rgba(255, 255, 255, .2)),
    color-stop(.25, transparent), color-stop(.5, transparent),
    color-stop(.5, rgba(255, 255, 255, .2)),
    color-stop(.75, rgba(255, 255, 255, .2)),
    color-stop(.75, transparent), to(transparent));
    content: "";
    position: absolute;
    top: 0;
    left: 0;
    bottom: 0;
    right: 0;
    background-size: 50px 50px;
    -webkit-animation: moveAnimate 2s linear infinite;
    overflow: hidden;
}
@-webkit-keyframes moveAnimate {
    0% {
       background-position: 0 0;
    }
    100% {
       background-position: 50px 50px;
    }
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/config/custom-target-cs1" %}

Width and height

The dimensions of the toast can be set using the width and height properties. This will individually set all toasts. You can create different custom dimension toasts.

By default, the toast can be rendered with `300px` width with `auto` height.

In mobile devices, the default width of the toast gets '100%' width of the page.

When you set toast width as '100%', the toast occupies full width and will be displayed at the top or bottom based on the position `Y` property.

Both the width and height properties allow setting pixels/numbers/percentage. The number value is considered as pixels.

APP.VUE

```
<template>
  <div id="app">
       <div id='container'>
        <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <ejs-toast ref='defaultRef' id='element' title='Matt sent you a
friend request' content='You have a new friend request yet to accept'
:position='position' width=400 height=150></ejs-toast>
            <div class='row' style="padding-top: 20px" id=
"toast_pos_target">
              <table>
                <tr>
                  <td>
                      <div style='padding:25px 0 0 0;'>
                          <ejs-radiobutton ref='topAlign' label='Top'
name='toast' value='Target' :change='checkboxChange'></ejs-radiobutton>
                      </div>
                  </td>
                  <td>
                      <div style='padding:25px 0 0 0;'>
                          <ejs-radiobutton ref='bottomAlign' label='Bottom'
checked=true name='toast' value='Global' :change='checkboxChange1'></ejs-
radiobutton>
                      </div>
                  </td>
                </tr>
                <tr>
                   <td>
                     <div style='padding:25px 0 0 0;'>
                          <ejs-checkbox label='100% Width'
:change='onChange'></ejs-checkbox>
                      </div>
                   </td>
                </tr>
              </table>
            </div>
        <br/><br/>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ToastPlugin } from '@syncfusion/ej2-vue-notifications';
import { ButtonPlugin, RadioButtonPlugin, CheckBoxPlugin, ChangeEventArgs as
CheckBoxChange, ChangeEventArgs } from '@syncfusion/ej2-vue-buttons';
Vue.use(ToastPlugin);
Vue.use(RadioButtonPlugin);
Vue.use(CheckBoxPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
    return {
        position: { X: "Center", Y: "Bottom" },
    }
  },
  mounted: function() {
      this.$refs.defaultRef.show();
      this.timeDelay = 500;
      this.toast = document.getElementById('element').ej2_instances[0];
  },
  methods: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      showBtnClick: function(args){
          this.$refs.defaultRef.show();
      },
      onChange: function(e) {
          if (e.checked) {
             this.toast.hide();
             this.toast.width = "100%";
             this.toast.title="";
             this.toast.content="<div class='e-custom'>Take a look at our
next generation <b>Javascript</b> <a
href='https://ej2.syncfusion.com/home/index.html' target='_blank'>LEARN
MORE</a></div>";
             this.toastShow(this.timeDelay);
          } else {
             this.toast.hide();
             this.toast.width= 300;
             this.toast.title= 'Matt sent you a friend request',
             this.toast.content='You have a new friend request yet to
accept',
             this.toastShow(this.timeDelay);
          }
      },
      checkboxChange: function(e) {
        if (e.event.target.checked) {
            this.toast.position.Y = "Top";
            this.toast.hide();
            this.toastShow(this.timeDelay);
        }
      },
      checkboxChange1: function(e) {
        if (e.event.target.checked) {
            this.toast.position.Y = "Bottom";
            this.toast.hide();
            this.toastShow(this.timeDelay);
        }
      },
      toastShow: function(timeOutDelay) {
         setTimeout(() => {
            this.toast.show();
        }, timeOutDelay);
      }
   }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
<style lang="scss">
#container {
  margin: 25px;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  top: 45%;
  width: 30%;
}
table td {
width: 50%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/config/width-height-cs1" %}

See Also

- Prevent duplicate toasts
- Customize the progress bar

Position in Vue Toast component

The toast position can be updated based on predefined positions or customizable positions. The predefined position combinations are updated in the X and Y position properties.

Predefined

X Positions

- Left
- Center
- Right

Y Positions

- Top
- Bottom

In multiple toast display, the new toast position will not be updated on dynamic change of property values until the old toast messages removed.

The toast occupies full width when you set the width to '100%', so the X positions will not affect the changes when the width is '100%'.

Custom

Custom X and Y positions can be given as pixels/numbers/percentage. The number value is considered as pixels based on the top and left values updated in the toast.

APP.VUE

```
<template>
   <div>
     <div class="col-lg-12 control-section toast-pos-section" style="max-
width: 300px;">
       <div class="e-sample-resize-container" id="toast_pos_target">
        <ejs-toast ref='toastRef' id='toast_pos' :position='position'
:target='target' title='Matt sent you a friend request' content='Hey, wanna
dress up as wizards and ride our hoverboards?' icon='e-laura'
:created='created'></ejs-toast>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <div id="toast_pos_target">
            <table style="width: 100%">
                <tbody><tr>
                    <td>
                        <div style="padding:25px 0 0 0;">
                            <ejs-radiobutton id='dropdownRadio'
label="Position" name="toastPos" value="Position" checked="true"
:change='checkboxChange2' ></ejs-radiobutton>
                        </div>
                    </td>
                    <td>
                        <div style="padding:25px 0 0 0;">
                         <ejs-radiobutton id='customRedio' label="Custom"
name="toastPos" value="Custom" :change='checkboxChange3'></ejs-radiobutton>
                         </div>
                    </td>
                </tr>
                <tr>
                    <td id="dropdownChoose" colspan="2">
                        <div id="dropdown" style="padding-top:25px;">
                         <ejs-dropdownlist ref='positionDropRef'
id='position' :dataSource='dropData' :fields='dropFields'
                        placeholder='Select a position'
:change='valueChange' :value='dropValue' popupHeight='200px' index=5></ejs-
dropdownlist>
                        </div>
                    </td>
                </tr>
                <tr>
                    <td colspan="2" id="customChoose" style="display: none">
                        <form id="formId" class="form-horizontal">
                            <div class="e-row">
                                <div class="e-float-input">
                                    <input class="e-input" id="xPos"
name="Digits" value="50" digits="true" data-digits-message="Please enter
digits only." required="">
                                    <span class="e-float-line"></span>
                                    <label class="e-float-text"
for="Digits">X Position</label>
                                </div>
                            </div>
                            <div class="e-row">
                                <div class="e-float-input">
                                    <input class="e-input" id="yPos"
name="Digits" value="50" digits="true" data-digits-message="Please enter
digits only." required="">
                                    <span class="e-float-line"></span>
                                    <label class="e-float-text"
for="Digits">Y Position</label>
                                </div>
                            </div>
                        </form>
                    </td>
                </tr>
                <tr>
                    <td>
                        <div style="padding:25px 0 0 0;">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                            <ejs-radiobutton id='radio1' label="Target"
name="toast" value="Target" :change='checkboxChange' ></ejs-radiobutton>
                        </div>
                    </td>
                    <td>
                        <div style="padding:25px 0 0 0;">
                            <ejs-radiobutton id='radio2' label="Global"
name="toast" value="Global" checked="true" :change='checkboxChange1' ></ejs-
radiobutton>
                        </div>
                    </td>
                </tr>
            </tbody></table>
            <div id="toast_btn" style="padding-top: 25px">
                <ejs-button ref='showButtonRef' class="e-btn e-control"
id="show_Toast" style="margin-right: 15px" v-
on:click.native="showClicked">Show Toast</ejs-button>
                <ejs-button class="e-btn e-control" id="hideTosat" v-
on:click.native="hideClicked">Hide All</ejs-button>
            </div>
        </div>
        </div>
    </div>
    </div>
    </template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
import { DropDownListPlugin, ChangeEventArgs } from '@syncfusion/ej2-vue-
dropdowns';
import { ButtonPlugin, RadioButtonPlugin, ChangeEventArgs as CheckBoxChange
} from '@syncfusion/ej2-vue-buttons';
import { isNullOrUndefined } from '@syncfusion/ej2-base';
Vue.use(ToastPlugin);
Vue.use(RadioButtonPlugin);
Vue.use(DropDownListPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
    return {
        position:  { X: 'Right', Y: 'Bottom' },
        target: document.body,
        dropData: [
            { Id: 'topleft', Text: 'Top Left' },
            { Id: 'topright', Text: 'Top Right' },
            { Id: 'topcenter', Text: 'Top Center' },
            { Id: 'topfullwidth', Text: 'Top Full Width' },
            { Id: 'bottomleft', Text: 'Bottom Left' },
            { Id: 'bottomright', Text: 'Bottom Right' },
            { Id: 'bottomcenter', Text: 'Bottom Center' },
            { Id: 'bottomfullwidth', Text: 'Bottom Full Width' }
        ],
        dropFields: { text: 'Text', value: 'Id' },
        dropValue: 'bottomright'
    }
  },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  mounted: function() {
      setTimeout(() => {
            this.toastShow(200);
        },200);
        this.initialWid = this.$refs.toastRef.ej2Instances.width.toString();
        this.customFlag = false;
        this.obj = document.getElementById("toast_pos").ej2_instances[0];
  },
  methods: {
      valueChange: function(e){
         this.$refs.toastRef.hide('All');
         this.setToastPosValue(e.value.toString());
         this.toastShow(1000);
       },
       toastShow: function(args){
         setTimeout(() => {
             this.$refs.toastRef.show();
            },args);
       },
       checkboxChange: function(args){
            var checkboxObj = args.event.target.ej2_instances[0];
            if (checkboxObj.checked) {
                this.$refs.toastRef.hide('All');
                this.obj.target = '#toast_pos_target';
                this.toastShow(1000);
            }
       },
       checkboxChange1: function(args){
          var checkboxObj = args.event.target.ej2_instances[0];
            if (checkboxObj.checked) {
                this.$refs.toastRef.hide('All');
                this.obj.target = document.body;
                this.toastShow(1000);
            }
       },
       checkboxChange2: function(args){
            var checkboxObj = args.event.target.ej2_instances[0];
            if (checkboxObj.checked) {
                this.$refs.toastRef.hide('All');
                document.getElementById('dropdownChoose').style.display =
'table-cell';
                document.getElementById('customChoose').style.display =
'none';

this.setToastPosValue(this.$refs.positionDropRef.value.toString());
                this.customFlag = false;
                this.toastShow(1000);
        }
       },
       checkboxChange3: function(args){
            var checkboxObj = args.event.target.ej2_instances[0];
            if (checkboxObj.checked) {
            this.$refs.toastRef.hide('All');
            document.getElementById('dropdownChoose').style.display =
'none';
            document.getElementById('customChoose').style.display = 'table-
cell';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            this.setcustomPosValue();
            this.customFlag = true;
            this.toastShow(1000);
         }
        },
        showClicked: function(args){
            if (this.customFlag) {
               this.setcustomPosValue();
         }
         this.$refs.toastRef.show();
        },
        hideClicked: function(args){
            this.$refs.toastRef.hide('All');
        },
         setcustomPosValue: function(args){
            this.obj.width = this.initialWid;
            this.obj.position.X =
parseInt(document.getElementById('xPos').value, 10);
            this.obj.position.Y =
parseInt(document.getElementById('yPos').value, 10);
        },
        setToastPosValue: function(value){
            this.obj.width = this.initialWid;
            switch (value) {
             case 'topleft':
                 this.obj.position.X = 'Left'; this.obj.position.Y = 'Top';
break;
             case 'topright':
                 this.obj.position.X = 'Right'; this.obj.position.Y = 'Top';
break;
             case 'topcenter':
                 this.obj.position.X = 'Center'; this.obj.position.Y = 'Top';
break;
             case 'topfullwidth':
                 this.obj.width = '100%'; this.obj.position.X = 'Center';
this.obj.position.Y = 'Top'; break;
             case 'bottomleft':
                 this.obj.position.X = 'Left'; this.obj.position.Y =
'Bottom'; break;
             case 'bottomright':
                 this.obj.position.X = 'Right'; this.obj.position.Y =
'Bottom'; break;
             case 'bottomcenter':
                 this.obj.position.X = 'Center'; this.obj.position.Y =
'Bottom'; break;
             case 'bottomfullwidth':
                 this.obj.width = '100%'; this.obj.position.X = 'Center';
this.obj.position.Y = 'Bottom'; break;
         }
        },
        created: function(args){
             document.addEventListener('click', function() {
               if (!isNullOrUndefined(this.$refs.toastRef) && event.target
!== this.$refs.showButtonRef.$el &&  this.$refs.toastRef.target ===
document.body) {
                    this.$refs.toastRef.hide('All');
                }
```

```
            }.bind(this));
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
<style lang="scss">
     .toast-pos-section #toast_pos_property {
        height: 500px;
        border: none;
        margin: auto;
    }
    #toast_pos_property td {
        width: 50%;
    }
    .e-toast-icon.e-laura.e-icons {
        border-radius: 50%;
        background-repeat: no-repeat;
        background-size: contain;
        background-image:
url(https://ej2.syncfusion.com/vue/demos/src/toast/resource/laura.png);
        height: 48px !important;
        margin: 0;
        width: 69px;
    }
    @media (min-width: 740px) {
        #toast_pos_property {
            width: 450px;
        }
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/positioning-cs1" %}

See Also

- Render toast with different positions
- Customize the progress bar

Action buttons in Vue Toast component

You can include action buttons to the toast component by adding the buttons property. The collection of Essential JS 2 button models can be bound to the model property inside the buttons property. You can also include the click event callback function for each button.

APP.VUE

```
<template>
   <div id='app'>
       <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
       <ejs-toast ref='elementToastTime' id='elementToastTime'
:position='position' title='Anjolie Stokes' content='<p><img
```

Copyright © 2001 -2024 Syncfusion Inc.

```
src="https://ej2.syncfusion.com/vue/demos/src/toast/resource/laura.png"></p>
' width=230 height=250 :buttons='button'></ejs-toast>
   </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
import { ButtonPlugin, RadioButtonPlugin, ChangeEventArgs as CheckBoxChange
} from '@syncfusion/ej2-vue-buttons';
import { closest } from '@syncfusion/ej2-base';
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
    return {
        position:  { X: 'Right', Y: 'Bottom' },
        button: [{
            model: { content: "Ignore" },
            click: this.btnClick
        }, {
            model: { content: "reply" }
        }]
    }
  },
  methods: {
      showBtnClick: function(args){
          this.$refs.elementToastTime.show();
      },
      btnClick: function(e){
          let toastEle = closest(e.target, '.e-toast');
          this.$refs.elementToastTime.hide(toastEle);
      }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
<style lang="scss">
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
#elementToastTime {
    text-align: center;
  }
  #elementToastTime img{
    width: 100px;
    height: 100px;
    border-radius: 50%;
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  #elementToastTime .e-toast-message {
    width: inherit;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/action-button-cs1" %}

See Also

- Configuring options

Timeout in Vue Toast component

The toast can be expired based on the timeOut property. The toast can live till the time out reaches without user interaction, a time out value is considered as a millisecond.

- The timeOut delay can be visually represented using Progress Bar.
- The extendedTimeOut property determines how long the toast should be displayed after a user hovers over it.

You can terminate the process by using the showCloseButton property for destroying the toast at any time.

APP.VUE

```
<template>
   <div id='app'>
       <div class="e-float-input"><input class="e-input"
id="toast_input_index" required="" value="0"><span class="e-float-
line"></span><label class="e-float-text">Enter timeOut</label></div>
       <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
       <ejs-toast ref='elementToastTime' id='elementToastTime'
:position='position' title='Anjolie Stokes' content='<p><img
src="https://ej2.syncfusion.com/vue/demos/src/toast/resource/laura.png"></p>
' width=230 height=250 :buttons='button'></ejs-toast>
   </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
    return {
        position:  { X: 'Right', Y: 'Bottom' },
        button: [{
            model: { content: "Ignore" }
        }, {
            model: { content: "reply" }
        }]
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
  methods: {
      showBtnClick: function(args){
          let value =
parseInt(document.getElementById('toast_input_index').value)
          this.$refs.elementToastTime.show({timeOut: value});
      }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
<style lang="scss">
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
#app {
    max-width: 200px;
}
#elementToastTime {
    text-align: center;
  }
  #elementToastTime img{
    width: 100px;
    height: 100px;
    border-radius: 50%;
  }
  #elementToastTime .e-toast-message {
    width: inherit;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/time-out/time-cs1" %}

Static toast

You can prevent auto hiding in a toast as visible like static by setting zero (0) value in the timeOut Property.

APP.VUE

```
<template>
   <div id='app'>
       <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
       <ejs-toast ref='element' id='element' :position='position'
title='Matt sent you a friend request' content='Hey, wanna dress up as
wizards and ride our hoverboards?' showCloseButton=true timeOut=0></ejs-
toast>
   </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
    return {
        position:  { X: "Right" }
    }
  },
  mounted: function() {
      this.$refs.element.show();
  },
  methods: {
      showBtnClick: function(args){
          this.$refs.element.show();
      }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
<style lang="scss">
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-laura {
    border-radius: 50%;
    background-image:
url('https://ej2.syncfusion.com/demos/src/toast/resource/laura.png');
    background-repeat: no-repeat;
    background-size: cover;
    height: 50px !important;
    width: 50px !important;
    margin-bottom: 2px;
    margin-top: auto;
    margin-bottom: auto;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/time-out/static-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

See Also

- Hide the toast on click

Template in Vue Toast component

The Template property in toast can be defined as HTML element, this can be either a string or selector.

The HTML element tag can be given as a string for the Template property.

`ts

template: " Toast Content "

`

The Template property allows getting the template content using the query selector. Here, the element ID attribute is specified in the template.

`ts

template: "#Template"

`

APP.VUE

```
<template>
   <div id='app'>
       <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
       <div id='element'></div>
       <script id="template_toast_ele" type="text/x-template">
    <div id='template_toast'>
        <div class="horizontal-align">
            <div class='toast-content'>
                <div class='toast-title'>
                    Weekend Alarm
                </div>
                <div class='toast-message'> With traffic, its likely to take
45 minutes to get to jenny's 24th Birthday Bash at Hillside Bar, 454 E.
                    Olive Way by 10:00PM </div>
            </div>
        </div>
        <img
src="https://ej2.syncfusion.com/demos/src/toast/resource/map.jpg"
width="100%" height="70%">
    </div>
</script>
   </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast, ToastOpenArgs, ToastCloseArgs,
ToastBeforeOpenArgs } from "@syncfusion/ej2-vue-notifications";
import { DropDownListPlugin, DropDownList, ChangeEventArgs } from
'@syncfusion/ej2-vue-dropdowns';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { compile, Browser, closest } from '@syncfusion/ej2-base';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
Vue.use(DropDownListPlugin);
export default {
  name: 'app',
  mounted: function() {
       this.toastObj = new Toast({
           template:
document.getElementById('template_toast_ele').innerHTML,
           position:  { X: 'Right', Y: 'Bottom' },
           target: document.body,
           extendedTimeout: 0,
           timeOut: 120000,
       });
      this.toastObj.appendTo('#element');
      this.toastObj.show();
  },
  methods: {
      showBtnClick: function(args){
          this.toastObj.show();
      }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
<style lang="scss">
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
#snooze,
#dismiss {
    background-color: #fff;
}
body>#element .e-toast {
    width: 400px !important;
}
.toast-template-section #reminder {
    text-align: center;
    margin: 15px;
}
#toast_custom .e-toast-template {
    display: inline-flex;
}
#template_toast .toast-icons {
    font-size: 35px;
    height: auto;
    margin: auto;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
#template_toast .toast-icons.e-alarm::before {
    content: "\e702";
}
#template_toast .horizontal-align {
    display: inline-flex;
    flex-direction: row;
    width: 100%;
}
#template_toast .horizontal-align,
#template_toast #snoozedropDown,
#template_toast .snooze,
#template_toast .snoozeBtn {
    margin: 10px 0;
}
#template_toast .horizontal-align .toast-content .toast-title {
    font-weight: 500;
}
#template_toast .horizontal-align .toast-content .toast-message {
    opacity: 0.4;
}
#template_toast .horizontal-align .toast-content {
    display: inline-flex;
    flex: 1;
    flex-direction: column;
    margin-left: 10px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/template-cs1" %}

See Also

- Add template dynamically

Animation in Vue Toast component

The toast component supports custom animations for both shows and hide actions from the provided animation option of the `Animation` library.

The default animation is given as `FadeIn` for showing the toast and `FadeOut` for hiding the toast.

The following sample demonstrates some types of animations that suit toast. You can check all the animation effects here.

APP.VUE

```
<template>
   <div id='app'>
        <div id="default" style="padding-bottom:75px;">
            <div class='row'>
                 <ejs-button ref='showButtonRef' class="e-btn"
id="show_toast" v-on:click.native="showBtnClick">Show Toast</ejs-button>
             </div>
            <div class='row'>
                <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
```

```
                    <label> Show Animation </label>
                </div>
                <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                     <ejs-dropdownlist ref='showAnimationRef'
id='ShowAnimation' :dataSource='animationData' :fields='easeFields'
                        placeholder='Select a animate type'
:change='showChange' :value='animationValue'></ejs-dropdownlist>
                </div>
            </div>
            <div class='row'>
                <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                    <label> Hide Animation </label>
                </div>
                <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                    <ejs-dropdownlist ref='hideAnimationRef'
id='HideAnimation' :dataSource='animationData' :fields='easeFields'
                        placeholder='Select a animate type'
:change='hideChange' :value='animationValue'></ejs-dropdownlist>
                </div>
            </div>
        </div>
         <ejs-toast ref='elementRef' id='element' :position='position'
:animation='animation' title='Matt sent you a friend request' content='You
have a new friend request yet to accept'></ejs-toast>
    </div>
</script>
</div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
import { DropDownListPlugin, DropDownList, ChangeEventArgs } from
'@syncfusion/ej2-vue-dropdowns';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
Vue.use(DropDownListPlugin);
export default {
  name: 'app',
   data: function(){
        return {
            position: { X: 'Right', Y: 'Bottom' },
            easeFields: { text: 'Text', value: 'Id' },
            animation: {
                show: { effect: "SlideRightIn" },
                hide: { effect: "SlideLeftOut" },
            },
            animationData: [
                { Id: 'FadeIn', Text: 'Fade In' },
                { Id: 'FadeZoomIn', Text: 'Fade Zoom In' },
                { Id: 'FadeZoomOut', Text: 'Fade Zoom Out' },
                { Id: 'FlipLeftDownIn', Text: 'Flip Left Down In' },
                { Id: 'FlipLeftDownOut', Text: 'Flip Left Down Out' },
                { Id: 'FlipLeftUpIn', Text: 'Flip Left Up In' },
                { Id: 'FlipLeftUpOut', Text: 'Flip Left Up Out' },
                { Id: 'FlipRightDownIn', Text: 'Flip Right Down In' },
                { Id: 'FlipRightDownOut', Text: 'Flip Right Down Out' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                { Id: 'FlipRightUpIn', Text: 'Flip Right Up In' },
                { Id: 'FlipRightUpOut', Text: 'Flip Right Up Out' },
                { Id: 'SlideBottomIn', Text: 'Slide Bottom In' },
                { Id: 'SlideBottomOut', Text: 'Slide Bottom Out' },
                { Id: 'SlideLeftIn', Text: 'Slide Left In' },
                { Id: 'SlideLeftOut', Text: 'Slide Left Out' },
                { Id: 'SlideRightIn', Text: 'Slide Right In' },
                { Id: 'SlideRightOut', Text: 'Slide Right Out' },
                { Id: 'SlideTopIn', Text: 'Slide Top In' },
                { Id: 'SlideTopOut', Text: 'Slide Top Out' },
                { Id: 'ZoomIn', Text: 'Zoom In' },
                { Id: 'ZoomOut', Text: 'Zoom Out' }
            ],
            animationValue: 'FadeIn'
        }
    },
  mounted: function() {
      this.toastObj = document.getElementById('element').ej2_instances[0];
      this.toastObj.show();
  },
  methods: {
      showBtnClick: function(args){
          this.toastObj.show();
      },
      showChange: function(args){
          this.toastObj.animation.show.effect = args.value;
      },
      hideChange: function(args){
          this.toastObj.animation.hide.effect = args.value;
      },
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
<style lang="scss">
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
#app {
    max-width: 200px;
}
.row {
    margin: 15px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/animations-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Toast services in Vue Toast component

The Vue Toast component provides a built-in utility function to render the toast with minimal code. The utility function will render the toast without the need of rendering the container element in the DOM where the toast is appended. So that, the toast can now be rendered on the go. The following are the option to render the toast using the utility function.

Show Toast with predefined types

The Toast component support 4 types of predefined toast with essential colors for various situations which can be shown using the ToastUtility.show by just defining the type of the toast without defining any class names. The following options are used as an argument on calling the utility function for predefined types:

| Options | Description |

|----------|-------------|

| content | Specifies the content that can be displayed on the Toast. |

| type | Specifies the type of the predefined Toasts. The 4 types of predefined toasts are Information, Success, Error, Warning |

| timeOut | Specifies the Toast display time duration on the page in milliseconds. Once the time expires, Toast message will be removed. Setting 0 as a time out value displays the Toast on the page until the user closes it manually. |

APP.VUE

```
<template>
   <div id='app'>
        <div>
            <ejs-button ref='infoButtonRef' class="e-btn e-control e-info"
id="info_toast" v-on:click.native="infoToast">Info Message</ejs-button>
            <ejs-button ref='successButtonRef' class="e-btn e-control e-
success" id="success_toast" v-on:click.native="successToast">Success
Message</ejs-button>
            <ejs-button ref='warningButtonRef' class="e-btn e-control e-
warning" id="warning_toast" v-on:click.native="warningToast">Warning
Message</ejs-button>
            <ejs-button ref='dangerButtonRef' class="e-btn e-control e-
danger" id="danger_toast" v-on:click.native="dangerToast">Danger
Message</ejs-button>
        </div>
        <br/>
        <div>
            <center><ejs-button ref='hideButtonRef' class="e-btn"
id="hide_toast" v-on:click.native="hideToast">Hide All</ejs-button></center>
        </div>
   </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast, ToastUtility } from "@syncfusion/ej2-vue-
notifications";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { closest } from '@syncfusion/ej2-base';
Vue.use(ToastPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(ButtonPlugin);
let ToastObj = undefined;
export default {
  name: 'app',
  data: function(){
    return { }
  },
  methods: {
        infoToast: function(){
            ToastObj = ToastUtility.show('Please read the comments
carefully', 'Information', 20000);
        },
        successToast: function(){
            toastObj = ToastUtility.show('Your message has been sent
successfully', 'Success', 20000);
        },
        warningToast: function(){
            toastObj = ToastUtility.show('There was a problem with your
network connection', 'Warning', 20000);
        },
        dangerToast: function(){
            toastObj = ToastUtility.show('A problem has been occurred while
submitting the data', 'Error', 20000);
        },
        hideToast: function(){
            ToastObj.hide('All');
        }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
<style lang="scss">
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/action-button-cs2" %}

Show Toast with ToastModel

The utility function can be called using the ToastModel as argument to show the toast where all the properties in the ToastModel like any events, position, close icon, action buttons, etc. can be used in the ToastUtility.show.

APP.VUE

```
<template>
   <div id='app'>
```

```
        <center><ejs-button ref='showButtonRef' class="e-btn" id="show_toast"
v-on:click.native="showToast">Show Toast</ejs-button></center>
    </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast, ToastUtility } from "@syncfusion/ej2-vue-
notifications";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { closest } from '@syncfusion/ej2-base';
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
let ToastObj = undefined;
export default {
  name: 'app',
  data: function(){
    return { }
  },
  methods: {
      showToast: function(args){
          ToastObj = ToastUtility.show({
            title: 'Toast Title',
            content: 'Toast shown using utility function with ToastModel',
            timeOut: 20000,
            position: { X: 'Right', Y: 'Bottom' },
            showCloseButton: true,
            click: this.toastClick,
            buttons:  [{
                model: { content: 'Close' }, click: this.toastClose
            }]
        });
      },
      toastClose: function(){
        ToastObj.hide();
      }
      toastClick: function(){
        console.log('Toast click event triggered');
      }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
<style lang="scss">
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/action-button-cs3" %}

Style in Vue Toast component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the toast title

Use the following CSS to customize the default toast's content properties like font-family, font-size and color.

```css
/* To change color, font family and font size */
.e-toast-container .e-toast .e-toast-message .e-toast-title {
color: red;
font-size: 18px;
font-weight: bold;
}
```

Customizing the toast content

Use the following CSS to customize the default toast's content properties like font-family, font-size and color.

```css
/* To change color, font family and font size */
.e-toast-container .e-toast .e-toast-message .e-toast-content {
color: aqua;
font-size: 13px;
font-weight: normal;
}
```

Customizing the toast icon

Use the following CSS to customize the default toast icon color.

```css
/* To change icon color */
.e-toast-container .e-toast .e-toast-icon {
color: yellow;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

Customizing the toast background

Use the following CSS to customize the default toast's background color.

`css

/ *To change background color* /

.e-toast-container .e-toast {

background-color: navy;

}

`

Accessibility in Vue Toast component

The Toast component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Toast component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/documentation/not-supported.png" alt="No"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

Copyright © 2001 -2024 Syncfusion Inc.

</style>

<img src="https://cdn.syncfusion.com/content/images/documentation/full.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/partial.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/documentation/not-supported.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The toast component has been designed with WAI-ARIA specifications in mind by applying the prompt WAI-ARIA roles, states, and properties with the keyboard support. It helps users who use assistive WAI-ARIA accessibility support, which is achieved using attributes.

It provides information about the elements in a document for assistive technology.

The toast component implements the keyboard navigation support by using the following WAI-ARIA practices and is tested in major screen readers.

<!-- markdownlint-disable MD033 -->

| Class | Description |

| -------- | -------- |

| role | alert: Identifies the element as a container when alert content will be added or updated. |

APP.VUE

```
<template>
   <div id='app'>
         <ejs-toast ref='elementRef' timeOut=0 id='element' title='Matt sent you a friend request' content='You have a new friend request yet to accept'></ejs-toast>
    </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
Vue.use(ToastPlugin);
export default {
  name: 'app',
  mounted: function() {
      this.$refs.elementRef.show();
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-notifications/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/accessibility-cs1" %}

Ensuring accessibility

The Toast component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Toast component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Toast component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/toast.html" %}

See also

- Accessibility in Syncfusion Vue components

How To

Prevent duplicate toast display in Vue Toast component

You can prevent identical same toast displaying in a screen by the event function and terminate the toast displaying process by setting the cancel event property in the beforeOpen event.

The following sample demonstrates preventing duplicate title toast element displaying with custom code blocks.

APP.VUE

```
<template>
   <div id='app'>
        <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
        <ejs-toast ref='elementRef' id='element' title='Sample Toast Title'
content='Sample Toast content' :position='position' :beforeOpen='beforeOpen'
:close="onClose" :created="onCreated"></ejs-toast>
   </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
        return {
            position: { X: 'Center' }
        }
   },
  mounted: function() {
      this.toastFlag = 0;
      this.toasts = [
                { title: 'Warning !', content: 'There was a problem with your
network connection.', isOpen: false },
                { title: 'Success !', content: 'Your message has been sent
successfully.', isOpen: false },
                { title: 'Error !', content: 'A problem has been occurred
while submitting your data.', isOpen: false }];
      this.$refs.elementRef.show(this.toasts[this.toastFlag]);
```

```
      ++this.toastFlag;
  },
  methods: {
       showBtnClick: function(args){
        this.$refs.elementRef.show(this.toasts[this.toastFlag]);
        ++this.toastFlag;
        if (this.toastFlag === (this.toasts.length)) {
            this.toastFlag = 0;
        }
       },
       beforeOpen: function(e){
        if (this.preventDuplicate(e)) {
            e.cancel = true;
        }
       },
       onCreated: function() {
        if (!isNullOrUndefined(this.toastInstance)){
            this.toastInstance.show(this.toasts[this.toastFlag]);
            ++this.toastFlag;
        }
        },
        onClose: function(e) {
            for (let i: number = 0; i < this.toasts.length; i++) {
                if (this.toasts[i].title === e.options.title) {
                    this.toasts[i].isOpen = false;
                }
            }
        },
        preventDuplicate: function(e){
           for (let i = 0; i < this.toasts.length; i++) {
            if (this.toasts[i].title === e.options.title &&
!this.toasts[i].isOpen) {
                this.toasts[i].isOpen = true;
                return false;
            }
           }
           return true;
       }
    }
}
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/how-to/duplicate-cs1" %}

Restrict the maximum toast to show in Vue Toast component

You can restrict the maximum toast count by using the event callback function and terminate the toast displaying process by setting the cancel event property in the beforeOpen event.

Copyright © 2001 -2024 Syncfusion Inc.

The following sample demonstrates restricting toast displaying up to 3. You can restrict by your own count with custom code blocks.

APP.VUE

```
<template>
   <div id='app'>
        <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
        <ejs-toast ref='elementRef' id='element' title='Sample Toast Title'
content='Sample Toast content' :position='position'
:beforeOpen='beforeOpen'></ejs-toast>
   </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
        return {
            position: { X: 'Right', Y: 'Bottom' }
        }
   },
  mounted: function() {
      this.toastFlag = 0;
      this.maxCount = 3;
      this.toasts = [
                { title: 'Warning !', content: 'There was a problem with your
network connection.' },
                { title: 'Success !', content: 'Your message has been sent
successfully.' },
                { title: 'Error !', content: 'A problem has been occurred
while submitting your data.' },
                { title: 'Information !', content: 'Please read the comments
carefully.' }];
      this.$refs.elementRef.show(this.toasts[this.toastFlag]);
      ++this.toastFlag;
  },
  methods: {
       showBtnClick: function(args){
           this.$refs.elementRef.show(this.toasts[this.toastFlag]);
           ++this.toastFlag;
           if (this.toastFlag === (this.toasts.length)) {
               this.toastFlag = 0;
           }
       },
       beforeOpen: function(e){
          if (this.maxCount ===
this.$refs.elementRef.ej2Instances.element.childElementCount) {
               e.cancel =true;
           } else {
               e.cancel = false;
           }
```

```
        }
    }
}
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css';
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/how-to/maximum-cs1" %}

Customize progress bar theme and sizing in Vue Toast component

By default, the progress bar appears based on the theme stylings and dimensions. You can customize the progress bar stylings using custom CSS or event functions.

The following sample demonstrates customizing the progress bar stylings using the beforeOpen event.

APP.VUE

```
<template>
   <div id='app'>
        <ejs-toast ref='elementRef' id='element' title='Matt sent you a
friend request' content='You have a new friend request yet to accept'
:position='position' :beforeOpen='beforeOpen' showProgressBar=true></ejs-
toast>
        <div class="row">
            <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
        </div>
        <div class="row" style="padding-top: 20px">
          <div class="e-float-input">
            <input class="e-input" id="progressHeight" name="Digits"
value="4" required>
            <span class="e-float-line"></span>
            <label class="e-float-text" for="Digits">Progress Bar
Height</label>
            </div>
        </div>
        <div class="row" style="padding-top: 20px">
            <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                <label> Progress Bar Color </label>
            </div>
            <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                <ejs-dropdownlist ref='ProgressRef' id='Progress'
:dataSource='data' :fields='dataFields'
                        placeholder='Select a animate type' index=0
popupHeight='150px' :change='valueChange'></ejs-dropdownlist>
            </div>
        </div>
   </div>
</template>
<script>
```

```
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
import { DropDownListPlugin } from "@syncfusion/ej2-vue-dropdowns";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ToastPlugin);
Vue.use(DropDownListPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
        return {
            position: { X: 'Right', Y: 'Bottom' },
            dataFields: { text: 'Text', value: 'Id' },
            data : [
                { Id: 'red', Text: 'Red' },
                { Id: 'cyan', Text: 'Cyan' },
                { Id: 'blue', Text: 'Blue' },
                { Id: 'yellow', Text: 'Yellow' },
                { Id: 'pink', Text: 'Pink' }
            ],
        }
   },
  mounted: function() {
      this.$refs.elementRef.show();
  },
  methods: {
       showBtnClick: function(args){
           this.$refs.elementRef.show();
       },
       beforeOpen: function(e){
          var progress = e.element.querySelector('.e-toast-progress');
          progress.style.height =
document.getElementById('progressHeight').value + 'px';
          progress.style.backgroundColor =
this.$refs.ProgressRef.ej2Instances.value;
       },
        valueChange: function(){
            var progressEles =
this.$refs.elementRef.ej2Instances.element.querySelectorAll('.e-toast-
progress');
            progressEles.forEach((ele)=> {
              ele.style.backgroundColor =
this.$refs.ProgressRef.ej2Instances.value;
            });
       },
    }
}
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css';
</style>
<style lang="scss">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/how-to/custom-progress-cs1" %}

Play an audio before open the toast in Vue Toast component

The following sample demonstrates how to play an audio in background while opening the toast by including audio play codes into the beforeOpen event function.

To stop the audio after displaying the toast, use the open event in toast. For further customization, check the Toast Events APIs.

APP.VUE

```
<template>
   <div id='app'>
       <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
        <ejs-toast ref='elementRef' id='element' title='Matt sent you a
friend request' content='You have a new friend request yet to accept'
:position='position' :beforeOpen='beforeOpen'></ejs-toast>
   </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
        return {
            position: { X: 'Right', Y: 'Bottom' }
        }
   },
  methods: {
       showBtnClick: function(args){
           this.$refs.elementRef.show();
       },
       beforeOpen: function(e){
          var audio = new
Audio('https://drive.google.com/uc?export=download&id=1M95VOpto1cQ4FQHzNBaLf
0WFQglrtWi7');
          audio.play();
       }
    }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css';
</style>
<style lang="scss">
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/how-to/audio-cs1" %}

Show different types of toast in Vue Toast component

The Essential JS 2 toast has the following predefined styles that can be defined using the cssClass property for achieving different types of toast:

| Class | Description |

| -------- | -------- |

| e-toast-success | Represents a positive toast |

| e-toast-info | Represents an informative toast |

| e-toast-warning | Represents a toast with caution |

| e-toast-danger | Represents a negative toast |

APP.VUE

```
<template>
   <div id='app'>
       <ejs-button ref='showButtonRef' class="e-btn" id="showToast" v-
on:click.native="showBtnClick">Show Types</ejs-button>
       <ejs-toast ref='elementRef' id='element' :position='position'></ejs-
toast>
   </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
        return {
            position: { X: 'Right', Y: 'Bottom' }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        }
    },
     mounted: function() {
     this.toastFlag = 0;
      this.toasts = [
                  { title: 'Warning !', content: 'There was a problem with
your network connection.', cssClass: 'e-toast-warning' },
                  { title: 'Success !', content: 'Your message has been sent
successfully.', cssClass: 'e-toast-success'},
                  { title: 'Error !', content: 'A problem has been occurred
while submitting your data.', cssClass: 'e-toast-danger' },
                  { title: 'Information !', content: 'Please read the comments
carefully.', cssClass: 'e-toast-info' }];
      this.$refs.elementRef.show(this.toasts[this.toastFlag]);
      ++this.toastFlag;
      },
  methods: {
        showBtnClick: function(args){
            this.$refs.elementRef.show(this.toasts[this.toastFlag]);
            ++this.toastFlag;
            if (this.toastFlag === (this.toasts.length)) {
               this.toastFlag = 0;
            }
        }
    }
}
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css';
</style>
<style lang="scss">
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/how-to/different-toast-cs1" %}

Show multiple toasts in various positions in Vue Toast component

By default, the positions of the new toasts are only updated after the visible toasts have been destroyed. If You need to display multiple toasts with different positions, initiate another toasts.

The following sample demonstrates adding multiple toasts in different positions.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
   <div id='app'>
       <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Right Position Toast</ejs-button>
       <ejs-button ref='showButton1Ref' class="e-btn" id="show_toast1" v-
on:click.native="showBtn1Click">Show Left Position Toast</ejs-button>
       <ejs-toast ref='elementRef' id='element' :position='position'
title='Warning !' content='There was a problem with your network
connection.' :click='onClick'></ejs-toast>
       <ejs-toast ref='element1Ref' id='element1' :position='position1'
title='Success !' content='Your message has been sent successfully.'
:click='onClick'></ejs-toast>
   </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
        return {
            position: { X: 'Right', Y: 'Bottom' },
            position1: { X: 'Left', Y: 'Bottom' },
        }
   },
    mounted: function() {
       this.$refs.elementRef.show();
       this.$refs.element1Ref.show();
     },
  methods: {
       showBtnClick: function(args){
           this.$refs.elementRef.show();
       },
       showBtn1Click: function(args){
           this.$refs.element1Ref.show();
       },
       onClick: function(e){
            e.clickToClose = true;
       }
    }
}
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css';
</style>
<style lang="scss">
#loader {
  color: #008cff;
  height: 40px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/how-to/position-cs1" %}

Close the toast with click tap in Vue Toast component

By default, the toasts are expired based on the timeOut value. You can customize the toast hiding process with click/tap action by setting the event args in the clicked callback function with static Toast.

APP.VUE

```
<template>
   <div id='app'>
       <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
       <ejs-toast ref='elementRef' id='element' :position='position'
title='Matt sent you a friend request' content='You have a new friend
request yet to accept' :click='onClick' timeOut=0></ejs-toast>
    </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast } from "@syncfusion/ej2-vue-notifications";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
        return {
            position:  { X: 'Right', Y: 'Bottom' }
        }
   },
    mounted: function() {
       this.$refs.elementRef.show();
     },
  methods: {
       showBtnClick: function(args){
           this.$refs.elementRef.show();
       },
       onClick: function(e){
            e.clickToClose = true;
       }
    }
}
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import '../../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css';
</style>
<style lang="scss">
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/how-to/close-cs1" %}

Add dynamic template in Vue Toast component

Toast supports to change templates dynamically with displaying in multiple toasts. You can change the toast properties while calling in the show method.

APP.VUE

```
<template>
   <div id='app'>
       <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
       <ejs-toast ref='elementRef' id='element' :position='position'
:click='onClick'></ejs-toast>
        <div id='templateToast' style="display: none;color:red"> System
affected by virus !!! </div>
    </div>
</template>
<script>
import Vue from "vue";
import { ToastPlugin, Toast, ToastClickEventArgs  } from "@syncfusion/ej2-
vue-notifications";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
        return {
            position:  { X: 'Right', Y: 'Bottom' }
        }
   },
    mounted: function() {
       this.toastFlag = 0;
       this.toasts = [
           { template: '2 Mail has received'},
           { template: 'User Guest Logged in'},
           { template: 'Logging in as Guest'},
           { template: 'Ticket has reserved '},
           { template: '#templateToast' }];
       this.$refs.elementRef.show(this.toasts[this.toastFlag]);
       ++this.toastFlag;
```

```
      },
  methods: {
        showBtnClick: function(args){
            this.$refs.elementRef.show(this.toasts[this.toastFlag]);
            ++this.toastFlag;
            if (this.toastFlag === (this.toasts.length)) {
               this.toastFlag = 0;
            }
        },
        onClick: function(e){
             e.clickToClose = true;
        }
    }
}
</script>
<style>
@import '../../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-popups/styles/material.css';
@import '../../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css';
</style>
<style lang="scss">
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/how-to/template-cs1" %}

Prevent toast close with mobile swipe in Vue Toast component

You can prevent the toast close with mobile swipe action by setting beforeClose argument cancel value to true while argument type as a swipe. The following code shows how to prevent toast close with mobile swipe.

The following sample demonstrates preventing toast close with mobile swipe element displaying with custom code blocks.

APP.VUE

```
<template>
  <div id="app">
        <ejs-button ref='showButtonRef' class="e-btn" id="show_toast" v-
on:click.native="showBtnClick">Show Toast</ejs-button>
        <ejs-toast ref='defaultRef' title='Matt sent you a friend request'
content='You have a new friend request yet to accept'
:beforeClose='onBeforeClose' :position='position'></ejs-toast>
  </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { ToastPlugin } from '@syncfusion/ej2-vue-notifications';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ToastPlugin);
Vue.use(ButtonPlugin);
export default {
  name: 'app',
  data: function(){
    return {
        position: { X: 'Center' }
    }
  },
  mounted: function() {
      this.$refs.defaultRef.show();
  },
  methods: {
      showBtnClick: function(){
          this.$refs.defaultRef.show();
      },
      onBeforeClose: function(e) {
        if (e.event.type === "swipe") {
            this.toast.cancel = true;
        }
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-
notifications/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/toast/how-to/preventSwipe-cs1" %}

Toolbar

Getting Started with the Vue Toolbar Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Toolbar component

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

Install the below required dependency package in order to use the Toolbar component in your application.

`javascript

|-- @syncfusion/ej2-vue-navigations

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-navigations

Copyright © 2001 -2024 Syncfusion Inc.

|-- @syncfusion/ej2-inputs

|-- @syncfusion/ej2-buttons

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

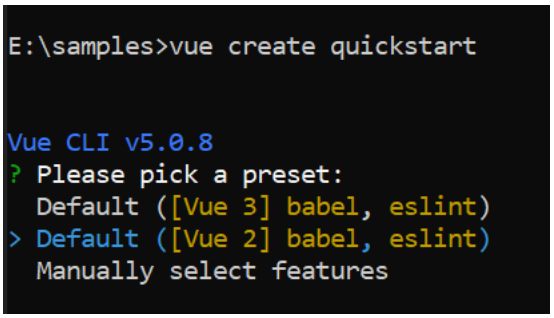
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Toolbar component as an example. Install the @syncfusion/ej2-vue-navigations package by running the following command:

`bash

npm install @syncfusion/ej2-vue-navigations --save

`

or

`bash

yarn add @syncfusion/ej2-vue-navigations

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Toolbar component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Toolbar component:

1\. First, import and register the Toolbar component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { ToolbarComponent, ItemDirective, ItemsDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
components: {
'ejs-toolbar': ToolbarComponent,
'e-item': ItemDirective,
'e-items': ItemsDirective
}
}
</script>
```

2\. In the template section, define the Toolbar component with text property.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-toolbar >
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-items>
<e-item text='Cut'></e-item>
<e-item text='Copy'></e-item>
<e-item text='Paste'></e-item>
<e-item type='Separator'></e-item>
<e-item text='Bold'></e-item>
<e-item text='Italic'></e-item>
<e-item text='Underline'></e-item>
</e-items>
</ejs-toolbar>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

~/SRC/APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:100%;">
        <br>
         <ejs-toolbar >
    <e-items>
             <e-item text='Cut'></e-item>
             <e-item text='Copy'></e-item>
             <e-item text='Paste'></e-item>
             <e-item type='Separator'></e-item>
             <e-item text='Bold'></e-item>
             <e-item text='Italic'></e-item>
             <e-item text='Underline'></e-item>
          </e-items>
    </ejs-toolbar>
    </div>
  </div>
</template>
<script>
import { ToolbarComponent, ItemDirective, ItemsDirective } from
"@syncfusion/ej2-vue-navigations";
export default {
  components: {
    'ejs-toolbar': ToolbarComponent,
    'e-item': ItemDirective,
    'e-items': ItemsDirective
  },
  name: 'app',
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/toolbar/getting-started-cs1" %}

Initialize the Toolbar using HTML elements

The Toolbar component can be rendered based on the given HTML element using <ejs-toolbar>. You need to follow the below structure of HTML elements to render the Toolbar inside the <ejs-toolbar> tag.

`

<ejs-toolbar> --> Root Toolbar Element

  --> Toolbar Items Container

  --> Toolbar Item Element

</ejs-toolbar>

`

~/SRC/APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:100%;">
        <br>
         <ejs-toolbar>
            <div>
                <div><button class="e-btn e-tbar-btn">Cut</button></div>
                <div><button class="e-btn e-tbar-btn">Copy</button></div>
                <div><button class="e-btn e-tbar-btn">Paste</button></div>
                <div class="e-separator"></div>
                <div><button class="e-btn e-tbar-btn">Bold</button></div>
                <div><button class="e-btn e-tbar-btn">Italic</button></div>
            </div>
          </ejs-toolbar>
    </div>
  </div>
</template>
<script>
import { ToolbarComponent } from "@syncfusion/ej2-vue-navigations";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  components: {
    'ejs-toolbar': ToolbarComponent
  },
  name: 'app',
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/toolbar/toolbar-container-cs1" %}

See Also

- How to add Toggle Button

Note: You can refer to our Vue Toolbar feature tour page for its groundbreaking feature representations. You can also explore our Vue Toolbar example that shows you how to render the Toolbar in Vue.

Getting Started with Syncfusion Vue Toolbar Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Toolbar component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Setup the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will creates a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion Vue Toolbar component to the project.

Adding Syncfusion Vue Toolbar package in the application

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Toolbar component as an example. To use the Vue Toolbar component in the project, the @syncfusion/ej2-vue-navigations package needs to be installed using the following command.

`bash

npm install @syncfusion/ej2-vue-navigations --save

`

or

`bash

yarn add @syncfusion/ej2-vue-navigations

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Toolbar component and its dependents were imported into the <style> section of the **src/App.vue** file.

`html

<style>

@import "../node_modules/@syncfusion/ej2-base/styles/material.css";

@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";

@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";

@import "../node_modules/@syncfusion/ej2-vue-navigations/styles/material.css";

</style>

`

Adding Syncfusion Vue Toolbar component in the application

Follow the below steps to add the Vue Toolbar component using Composition API:

Copyright © 2001 -2024 Syncfusion Inc.

1.First, import and register the Toolbar component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { ToolbarComponent as EjsToolbar, ItemsDirective as EItems,
ItemDirective as EItem } from "@syncfusion/ej2-vue-navigations";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import {
ToolbarComponent,
ItemDirective,
ItemsDirective,
} from "@syncfusion/ej2-vue-navigations";
export default {
name: "App",
components: {
"ejs-toolbar": ToolbarComponent,
"e-items": ItemsDirective,
"e-item": ItemDirective
}
}
</script>
```

2.Add the component definition in template section.

`html

<template>

<ejs-toolbar id="toolbar" width="800">

<e-items>

<e-item text="Cut" prefixIcon="e-cut-icon tb-icons"></e-item>

<e-item text="Copy" prefixIcon="e-copy-icon tb-icons"></e-item>

<e-item type="Separator"></e-item>

<e-item text="Bold" prefixIcon="e-bold-icon tb-icons"></e-item>

<e-item text="Underline" prefixIcon="e-underline-icon tb-icons"></e-item>

<e-item text="Italic" prefixIcon="e-italic-icon tb-icons"></e-item>

<e-item text="Color-Picker" prefixIcon="e-color-icon tb-icons"></e-item>

<e-item type="Separator"></e-item>

<e-item text="A-Z Sort" prefixIcon="e-ascending-icon tb-icons"></e-item>

<e-item text="Z-A Sort" prefixIcon="e-descending-icon tb-icons"></e-item>

Copyright © 2001 -2024 Syncfusion Inc.

<e-item text="Clear" prefixIcon="e-descending-icon tb-icons"></e-item>

</e-items>

</ejs-toolbar>

</template>

`

3.Add styles for Toolbar component to render icons.

`html

<style>

.e-tbar-btn .e-icons {

font-family: "Material_toolbar";

font-size: 16px;

font-style: normal;

font-weight: normal;

font-variant: normal;

text-transform: none;

}

.custom_bold .e-tbar-btn-text {

font-weight: 900;

}

.custom_italic .e-tbar-btn-text {

font-style: italic;

}

.custom_underline .e-tbar-btn-text {

text-decoration: underline red;

}

.e-txt-casing .e-tbar-btn-text {

font-variant: small-caps;

}

@font-face {

font-family: "Material_toolbar";

src: url(data:application/x-font-ttf;charset=utf-8base64,AAEAAAAKAIAAAwAgT1MvMj1tShMAAAEoAAAAVmNtYXDoMOjqAAACDAAAAHhnbHlmIuy19QAAAswAACNMaGVhZA6okZMAAADQAAAANmhoZWEIUQQkAAAArAAAACRobXR4jAAAAAAAAYAAAACMbG9jYYc0kUIAAAKEAAAASG1heHABOwG8AAABCAAAACBuYW1lx/RZbQAAJhgAAAKRcG9zdJZeEVUAACi

sAAACGAABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAIwABAAAAAQAAAQsu/F8PPP
UACwQAAAAAANXLJlEAAAAA1csmUQAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAjAbAADgAAAAAAg
AAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnIQQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAA
AAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEA
AAABAAAAAQAAAAEAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAGQAAAAEAAQAA
QAA5yH//wAA5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcACAAJAAoACwAMAA0ADgAPABAAEQAS
ABMAFAAVABYAFwAYABkAGgAbABwAHQAeAB8AIAAhACIAAAAAADIAjgFwAfgCIAKYAxIDSAO2BRYFMAV
cBnIGugb2ByoHQgguCNYJRgn6CiQKiAquCsgMFgzADOYNzg7WDvAQyBEyEaYABwAAAAAD9APzAAMABw
AKAA4AEgAVABkAADchNSElTUhJTkBBSE1ITUhNSEFFxEnITUhDAPo/BgBtgIy/c7+SgG2AjL9zgIy/c7+Svr6A
+j8GAxefV67Pl19Xvr6AfScXgAAAAIAAAAAA/QD9AAEAEgAACUhNxc3AREfDyE/DxEvDyEPDgOF/PbDisP9g
QEBAwQEBgYICAgJCgoLCwsDCgsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQoKCwsL/PYLCwsKCgkIC
AgGBgQEAwGz+qf6AYX89gsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQoKCwsLAwoLCwsKCgkICAgG
BgQEAwEBAQEDBAQGBggICAkKCgsLAAACAAAAAAPzA/QAQAC/AAABFQ8PLw8/Dx8OAQ8ELwErAQ8FFR
8FBxcPAxUfBzsBNx8LOwI/Cx8BOwE/Bj0BLwQ/Aic/BC8HKwEHLwsrAg8FArIBAgUGBwkKDAwODxAQERITE
hIREQ8PDgwMCgkHBgUCAQECBQYHCQoMDA4PDxEREhITEhEQEA8ODAwKCQcGBQL+zxUWFhUWfwUFB
AUDBANqAgEBAgIDbgMDbwMCAQEBAmkDBAQEBQSEFBYWFxQCAgIDBAQEBcwFBAQEAwICAhQXFRYVg
AQFBQQEAwNoAgEBAgIDcAEBAQNvAgIBAQEBA2gDBAQFBAWDFBYWFxIBAgMDAwQFBcwFBAQDBAICA
gAJCRIQEBAODgwLCgkHBgQDAQEDBAYHCQoLDA4OEBAQEhISEhAQEA4ODAsKCQcGBAMBAQMEBgcJCgs
MDg4QEBASAc6ECwwNDjIBAQICA7QEBQQFBAMEUjIyVgMEBAQFBAWwAwICATMODQwLhAQEBAMCAg
ECAgIDBAMEhAsMDQ4yAQECAgOwBAQFBAUEAwRSDAwaMlYDBAQEBQQFsAMCAgEzDg0MC4QEAwQD
AgICAgICAwQDAAAAAAMAAAAA/MD2AAyADUAaQAAJRUfDTsBPw41LwgPBwMhAScXAQ8GHQEfBQEf
BjsBPwYBPwYvBwEDFgIDBAQGBgcICQkKCgsLCwsLCwoKCQgICAYGBAQDAQEDBAcMCQoLFCMtFQoJCQcF
BHv96gEL04X+4gYFBAQCAgICAgIEBAUBNwcHBwgHCAcIBwgHCAgHBwYBOAUEAwMCAQEBAQIDAwQFBv
4PlwsLCwoKCQkIBwYGBAQDAgIDBAQGBgcICQkKCgsLCwcPEBAYEBAPHCk3HRAQEBAQEAEIAQrThf7iBgcI
BwgHCAgICAgHBwcH/skGBQQEAwIBAQIDBAQFBgE3BwcHBwgICAgHCAgHCAcGAfEABQAAAAAD9APzAA
MABwALAA8AEwAANyE1ITchNSEnITUhNyE1ISchNSEMA+j8GN4CLP3U3gPo/BjeAiz91N4D6PwYDF6AW5x
efVqAXgAAAAAEAAAAAAP0A/QACQATABcAWwAAAQcVMzcXMzUjLwEjFTMbATM1IwElESERBxEfDyE/Dx
EvDyEPDgFro8ObnnROxOp0nZvqTij+8AGW/NReAQEDBAQGBggICAkKCgsLCwMKCwsLCgoJCAgIBgYEBAM
BAQEBAwQEBgYICAgJCgoLCwv89gsLCwoKCQgICAYGBAQDAQENAyOWliO4BSUBK/7VJQFSffzUAywR/PYL
CwsKCgkICAgGBgQEAwEBAQEDBAQGBggICAkKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJ
CgoLCwAAAAACAAAAAAOWA/QAAwBpAAA3ITUhExUfHTsBPx01ESMRDw8vDxEjagMs/NRKAgIDAwUFBg
cHCAkJCgsLCwwNDQ0ODw4PEA8QERARERERERBEQDxAPDg8ODQ0NDAsLCwoJCQgHBwYFBQMDAgKLAQ
MFBggKCwwODxARERMTFBQTExEREA8ODAsFCQcGBAKLDH0BsBERERERQEA8QDg8ODg0MDQsLCwoJCQg
HBwYFBQQDAgEBAgMEBQUGBggICQkKCgsMDAwNDg4ODw8PEBAQERERAb7+RRQTEhIREA8NDQsKCAY
FAwEBAwUGCAoLDQ0PCBASEhMTAcUABQAAAAAD9APXAAIABQANABcAGgAAJTcjASM3ATM3MxczAyM
FIQEVITUhATUhJTMnAgJx4wG/vl/+/Fot+i1a3FD9RgEg/t4Bov7UAST+aAF/6XQobQET//47eHgCM07+XD5
NAaQ/UHMAAAAAAQAAAAAD9ALoAF8AABMhJz8PHxo3Lx8PDycMAbWyDQ0ODg8PDxAQEBERERIREhA
QEBAQDw8PDw4ODg0NDQwMFxYTEhAHBgYGBXUHBwgJCQoLCwwNDQ0PDg8QEBERERITEhMUExQVFB
UVFRgYFxcXFhYVFhQUFBMTEhGwARi6CwsJCggICAYGBgQEAwIBAQEBAgIDBAQFBQYHBwgICQkKFRYYGh
sODg8PDygUFBMTEhISERARDw8PDg0NDAsLCgoICAgGBgQEAwMBAQECAwQFBgcJCQoLDA0ODg+6AAYA
AAAAA/MD9AA/AGsAqwDrAO8BMwAAARUfDTsBPw09AS8ODw4lHwk7AT8IPQEvByMnByMPByUfDz8PL
w8PDiUfDz8OPQEvDSsBDw0lESERBxEfDyE/DxEvDyEPDgHhAgMFBQYHCAkKCgsLDA0NDA0MCwsKCgkIBw
YFBQMCAgMFBQYHCAkKCgsLDA0MDQ0MCwsKCgkIBwYFBQMC/scBAQEFBwgKCwYGBwYGBgwKCAcFAQ
EBAQUHCAoMBgYGBwYGCwoIBwUBAQHzAQECBAQEBgYGCAcICQkJCgoJCQgJBwgGBgYEBAMDAQEBAQ
MDBAQGBgYIBwkICQkKCgkJCQgHCAYGBgQEBAIB/qgBAQMEBAYGBwgICQoKCgsLCwsLCgkJCQcHBwUFA
wMCAgMDBQUHBwcJCQkKCwsLCwsKCgoJCAgHBgQUHBwcJCQkKCwsLCwsKCgoJCAgHBgYEBAMBAlD81F4BAQMDBAUGBgcHCAkJCQkKAyYKCQ

kJCQgHBwYGBQQEAgEBAQECBAQFBgYHBwgJCQkJCvzaCgkJCQkIBwcGBgUEAwMBAWQNDAwMCwoKCQ
gHBgUFAwICAwUFBgcICQoKCwwMDA0NDAwLCwsJCQgHBwUEAwIBAQIDBAUHBwgJCQsLCwwMMQYGB
gsKCQcEAgEBAgQHCQoLBgYGBwYGCwoJBgUCAQECBQYJCgsGBvMJCgkICAgHBwYFBQQDAwEBAQEDAw
QFBQYHBwgICAkKCQoJCQkICAcGBwUFBAMCAQEBAQIDBAUFBwYHCAgJCQkGCwsKCgoJCAgHBgYEBAMB
AQEBAwQEBgYHCAgJCgoKCwsLCwsKCQkJBwcHBQUDAwICAwMFBQcHBwkJCQoLC9/81AMsA/zaCgkJCQk
IBwcGBgUEBAIBAQEBAgQEBQUHBwcICQkJCQoDJgoJCQkJCAcHBwUFBAQCAQEBAQIEBAUFBwcHCAkJCQ
kAAAAAAgAAAAADtQP0AAMACgAANyE1IQEjCQEjESFKA2z8lAEG7AGcAZzs/qAMfQIK/mQBnAFhAAYAAA
AAA/QD8wADAAcACwAPABIAFgAANyE1ISUhNSE1ITUhNSE1KQERNwMhNSEMA+j8GAG2AjL9zgIy/c4CMv
3O/kr6+gPo/BgMXn1efV19Xv4M+gGWXgAFAAAAAAPzA/MAJQBpAKgArADwAAABFT8bIw8GBR8PNSMv
DT8CJw8OHw8TByMPDBc/AzsBHwUzHwYzLxcjJREhEQcRHw8hPw8RLw8hDw4CKg8QDw8ODg4NDQwM
DAwKCwoKCQgJDw0KCQQDAgLxBAUGBggJC/7WDQ0NDg4PDw8PEA8QEBAQEAoKCQkJCQgIBgQEBAUD
AQEDBKsKCQgIBwYGBQUDAwMBAQEBAQEDAwQEBQYHBwgICgoL9g4OHA4ODQ4NDg0NDQwNDKkLDA
8HCAkJCAgICA4DEAUFBQMEAu4EBwkKDQ8REwoKCwsMDAwNDQ4ODg4PDy8KAZP81F4BAQMDBAUGB
gcHCAkJCQkKAyYKCQkJCQgHBwYGBQQEAgEBAQECBAQFBgYHBwgJCQkJCvzaCgkJCQkIBwcGBgUEAwMB
AZz0AgMDBAQFBQYHBwgICQkJCgsLCwsYGhsbDw4PDwoKCQgIBwaUDAsLCgkIBwcGBQQEAgIBAfEBAwM
EBQYIBQcGBw8PEA8ODa0NDQ0ODg4ODw8PDw8PEA8PEA8PEA8ODw8ODg0ODQ0MAj8BBAMDBAQFBg
cHBwkJCqoGBQQBAQICBAMJEAcHCAgJCh0dHRsaGRcXCgoKCQgICAcGBgYEBQMDBj781AMsA/zaCgkJCQ
kIBwcGBgUEAwMBAQEBAgQEBQYGBwcICQkJCQoDJgoJCQkJCAcHBgYFBAQCAQEBAQIEBAUFBwcHCAkJC
QkAAgAAAAAD8wPrAB8AMwAAEw8HHww/BBUhNSEBNwkBPwcvCDYKCAcGBQMCAQECAwUGBwgKrwg
JCgoKCgkINQKQ/Xv+20EBPAGOCgkHBgUDAgEBAgMFBgcJCtMBoQsMDQ0NDg4ODg0ODQ0NDAuvBgUD
AQECBAY0I14BJEH+xAGRCwwMDQ0ODg4ODg4NDQwMC9QABgAAAAAD9AP0AAMADwATAB0AIQAnAA
AlITUhIzMVIxUzFSMVMzUjNyE1ISMzBxUzNSM3NSM3ITUhJzMVMzUjAQYC7v0S+n0/P328vPoC7v0S+nh4
vHh4vPoC7v0S+j4/fWpeID4gPvrbXXw/P3w/u14gvPoAAAAABQAAAAAD9APbAAIABQANABcAGgAAJTcjAy
M3ATM3MxczAyMFIQEVITUhATUhJzMnAgJx4x6+Xv76Wi39LF3fTwFlAST+3AGk/tIBJP5mw+l0JXMBGP/+
N3h4AjRQ/lo+TQGpPk5zAAABAAAAAAOsA/QACwAAATMDIxUhNSMTMzUhAXGd88gCPJ3zyP3EAx79xNb
WAjzWAAAGAAAAAAP0A9QAAwBDAEcAhwCLAMsAACUhNSEHFR8OPw49AS8ODw4TITUhBxUfDTsBPw0
9AS8ODw4TITUhBxUfDj8OPQEvDg8OAQYC7v0S+glBAwMEBQUFBgcGCAcICAgICAcHBgYGBQQEAwMCAQ
ECAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAvoC7v0S+glBAwMEBQUFBgcGCAcICAgICAcHBgYG
BQQEAwMCAQECAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAvoC7v0S+glBAwMEBQUFBgcGCAcI
CAgICAcHBgYGBQQEAwMCAQECAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAkpeLwgIBwcHBwYF
BQUDBAICAQEBAQICBAMFBQUGBwcHBwgICAgIBwcGBgYFBAQDAwIBAQEBAgMDBAQFBgYGBwcICAFgX
S4ICAgHBwYGBgUEBAMDAgEBAgMDBAQFBgYGBwcICAgICAcHBwcGBQUFAwQCAgEBAQECAgQDBQUFB
gcHBwcIAUBdLggICAcHBgYGBQQEAwMCAQEBAQIDAwQEBQYGBgcHCAgICAgHBwcHBgUFBQMEAgIBAQ
EBAgIEAwUFBQYHBwcHCAAAAwAAAAADmQP0AAcAKACNAAABFSE1MxEhESUHFQ8GLwc/Bx8GJysBDw0
VERUfDTMhMz8NNRE1Lw0rAS8OKwEPDQEdAcZb/YQBbAEDBAYHBwkJCQkHBwYEAwEBAwQGBwcJCQkJ
BwcGBAOsvwkJCQgICAcGBgYEBAMCAgICAwQEBgYGBwgICAkJCQ8CQkJCAgIBwYGBgQEAwICAgIDBAQG
BgYHCAgICQkJvwMFBQYGBwgICQkJCgoKCwsLCwoKCgkJCQgIBwYGBgUDPoiI/SkC1y4FBQgIBwUEAwEBA
wQFBwgICQkJCAcFBQIBAQIFBQcICCQCAgMEBAYGBgcICAgJCQn9KQkJCQgICAcGBgUFBAMCAgICAwQFBQ
YGBwgICAkJCQLXCQkJCAgIBwYGBgQEAwICCgkJCAgIBwYGBQQEAwICAgIDBAQFBgYHCAgICQkAAQAAAAA
D9ALoAGAAAAExLw8PHxc/Gh8PByERA0QREhMTFBQUFRYWFhcXFxgYFRUVFBUUExQTEhMSERERERBAPD
g8NDQ0MCwsKCQkIBwd1BQYGBgcQEhMWFwwMDQ0NDg4ODw8PDxAQEBQEREQERAQDw8PDg
4NDbIBtQIuDw4ODQwLCgkJBwYFBAMCAQEBAwMEBAYGCAgICgoLCwwNDQ4PDw8REBESEhITExQUKA8
PDw4OGxoYFhUKCQkICAcHBgYFBAMCAQEBAwMEBAYGCAgICgoLCwwNDQ4PDw8REBESEhITExQUKA8
PDw4OGxoYFhUKCQkICAcHBgYGBQQEAwMCAQEBAwMEBAYGCAgICgoLCwwNDQ4PDw8REBESEhITExQUKA8
PDw4OGxoYFhUKCQkICAcHBgYGBggJCwu6AdAAAAAOAAAAAAP0A/MA
AgAFAAgACwAQABQAFwAbAB4AIQApAC0AMQB1AAABETclFzUXNyMFNyETFQUhEQEhJRMlMycFMSEnB
zcnBxcRBRMDBSUDEy0BEQMlIwUDEQcRHw8hPw8RLw8hDw4CGcj+ZaG3MJb+wM7+4jQBCv6EAy7+ggEK
dP1S3JkBCwEjWemWlvrIATJ0dP7n/up3dwEWAZhy/vQ0/vZyXgEBAwQEBgYICAgJCgoLCwsDCgsLCwoKCQ
gICAYGBAQDAQEBAQMEBAYGCAgICQoKCwL/PYLCwsKCgkICAgGBgQEAwEBxv7fWSQ730Bky8v+9QNxAY
H+f28BHx2ZmcukmTgJywEeP/7n/ud3dwEZARl3Bv5xAR1ycv7yAYAR/PYLCwsKCgkICAcHBQUEAwEBAQED

BAUFBwcICAkKCgsLCwMKCwsLCgoJCAgHBwUFBAMBAQEBAwQEBgYIBwkJCgoLCwAAAAAFAAAAAAP0A/
MAAwAHAAsADwATAAA3ITUhJSE1ISUhNSElITUhJSE1IQwD6PwYAVgCkP1w/qgD6PwYAVgCkP1w/qgD6P
wYDF6AW5xefV19XgAAAAAKAAAAAAP0A/MAAwAHAAsADwATABcAGwAfACMARwAAARUjNSMVIzUjFS
M1ARUjNSMVIzUjFSM1JRUjNSMVIzUjFSM1JxEfByE/BxEvByEPBgOW+j7bP9oDLPo+2z/aAyz6Pts/2l4BAw
UGAwgJCgOJCgkJBwYDBAIBAwUGAwgJCvx3CgkJBwYFAwElvb27u7u7ARrb29vb29v6vLy8vLy8hvyCCwoJB
wQGBAIBAwUHBwUJCgOECwoJBwQGBAIBAwUGCAkKAAAAAAUAAAAAA/QD8wADAAcACwAPABMAADc
hNSE1ITUhNSE1ITUhDAPo/BgCkP1wA+j8GAKQ/XAD6PwYDF6BV59efVqAXgAAAADAAAAAAP
0A00AAwAHAAsAADchNSE1ITUhNSE1IQwD6PwYA+j8GAPo/Bizb6Zwpm8AAAABQAAAAD9AP0AD8AX
wCfAKQBIgAAJQ8PLw8/Dx8OExUPBSsBLwU9AT8FOwEfBQMPDy8PPw8fEAE1IwUVHw8zPwMXBy8FDw8f
Dz8PNS8DNwEzNQE/BS8PDw4BOAEBAwMEBQYGBwgICQkKCgoKCgoJCQgIBwYGBQQDAwEBAQEDAwQF
BgYHCAgJCQoKCgoKCgkJCAgHBgYFBAMDAeICAgMDBQUFBQUFAwMCAgICAwMFBQUFBQUDAwIC4QEB
AwMEBQYGBwgICQkKCgoKCgoJCQgIBwYGBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAgHBgYFBA
MDAftkAV6W/K4BAwUHCAoMDQ4PERETExQUCwsVFBN2dgkKCgoVFhQUExMREQ8ODQwKCAcFAwEBA
wUHCAoMDQ4PERETExQUFBQTExERDw4NDAoIBwUDAQEEBgd2AV6W/ZYFBAMCAwEBAwUHCAoMDQ4
PERETExQUFBQTExERDw4NDAoIBwUD1AoKCgkJCAgHBgYFBAMDAQEBAQMDBAUGBgcICAkJCgoKCgoKC
QkICAcGBgUEAwMBAQEBAwMEBQYGBwgICQkKCgEiBQUFAwMCAgICAwMFBQUFBQUDAwICAgIDAwUF
AScKCgoJCQgIBwYGBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAgHBgYFBAMDAQEBAQMDBAUG
BgcICAkJCgqgZAFeMpYKChQTExERDw4NDAoIBwUDAQEEBgd2dgUEAwIDAQEDBQcICgwNDg8RERMTFBQ
UFBMTEREPDg0MCggHBQMBAQMFBwgKDA0ODxEREBxMUFAsLFRQTdv6iMgJqCQoKChUWFBQTExERDw
4NDAoIBwUDAQEDBQcICgwNDg8RERMTFAADAAAAAANXA7UAIgBFAJMAAAEzHw4PDisBNRMzHw4PDis
BNQMhPxEvDz8PLxghAkgKCgkJCAgHBwYGBAQEAgEBAQEDAwQFBgYHBwgJCAkKCeDACgoJCQgIBwcGBg
QEBAIBAQEBAgQEBAYGBwcICAkJCgrAwAHDDQwMDBcWFRMSEQ8NDAoHBgQBAQIDBAYHBwkKCgsNDA
4ODwsLCgoKCAgIBgYFBQMDAQEBAQECAwQEBAUGDA8QEhQVFgwMDA0NDQ0N/nABogICAwQEBgYGB
wgICQkKCQoKCQgJBwgGBgUFBAMCArsBdwICAwQEBgYGCAcICQkKCQoKCQkIBwgGBgYEBAMCArv9MQE
BAQIGCAoMDg8REhQUFhcYGBERERAQEA4ODgwMDAoJCQcICQkKCgoLDAsMDAwMDQwNDQwNDQwM
CwwLCxQUERAODQoFAwQDAgEBAQAABQAAAAAD9APzAAMABwALAA8AEwAANyE1ITUhNSE1ITUhNSE
1ITUhNSEMA+j8GAPo/BgD6PwYA+j8GAPo/BgMXn1enF59XX1eAAAAAAEAAAAAA9QD1ADUAAATHx8/Dx
cRIRcPDy8fPx8fDzMvHw8eKwECAwQFBggICQoMDA0ODhAQERISExQUFRUWFhcXGBgYGBgXFxcWFhUVF
BQTEhIREIr+ZrsMDA0ODg4PEBAQEBESERISEhIREhEQEQ8QDw8ODg0NDAwLCgoICQkKCgsMDA0ODg8QEBERERISEhMTFBMUFRQYGBgXFxYWFRUUFBMSEhEQEA4ODQ0LCgoICAYGBgUFBAQDAgIBAQAAAAAA

AgMDBAUFBgYHCAcICQgJCQgHCAcGBgUFBAMDAgEBvQECAwQEBAYGBgcHCAgICQkICAgHBwcFBgQFAw
MCAQECAwMFBAYFBwcHCAgICQkICAgHBwYGBgQEBAMCAf7qAQIDAwUEBgUHBwcICAgJCQgICAcHBgYG
BAQEAwIBAQIDBAQEBgYGBwcICAgJCQgICAcHBwUGBAUDAwIB/kQBAgMEBgcHCQsLDA0ODw8RERITFB
QVFhYXFxcZGBkZGgkICAgHBwYGBgQEBAMCAQECAwMEBAoEBAMDAgECAgIEBAUFBgYHBwgICAlkDg8N
Dg0ODA0MDAwLCwsKCQoICQcIBgYGBQQEAwMCAQECAwQGBwcJCwsMDQ4PDxEREhMUFBUWFhcXFxk
YGRkaGhkZGBkXFxcWFhUUFBMSEREPDw4NDAsLCQcHBgQDAgJTCAkICAcHBgYFBQQEAgICAgICBAQFBQY
GBwcICAkICQgJBwgGBwYFBQQDAwIBAQEBAgMDBAUFBgcGCAcJCAkICQgIBwcGBgUFBAQCAgICAgIEBAU
FBgYHBwgICQgJCAkHCAYHBgUFBAMDAgEBAQECAwMEBQUGBwYIBwkI1gkJCAcIBgcGBQUEAwMCAQEB
AQIDAwQFBQYHBggHCAkJCAkICAcHBgYFBQQEAgIBAQEBAgIEBAUFBgYHBwgICQgJCQgHCAYHBgUFBAM
DAgEBAQECAwMEBQUGBwYIBwgJCQgJCAgHBwYGBQUEBAICAQEBAQICBAQFBQYGBwcICAn+xhoZGRgZF
xcXFhYVFBQTEhERDw8ODQwLCwkHBwYEAwIBAgICBAQFBQYGBwcICAkICAgIBwcGBgsGBwYIBwgICQkIB
wgGBwYFBQQDAwIBAQECAgMEBQUFBgcHCAgJCQoKCwoMCwwNDA0NDg0ODg4XFxYWFRUVFBQTExIR
ERAPDw4NDQsLCgkIBwYFBAMBAQECAwQGBwcJCwsMDQ4PDxEREhMUFBUWFhcXFxkYGRkAAgAAAAAD
9AO1AAgAVAAAARchFSEHFzcnJREVHw4hPw49ASMVIREhFTM9AS8OIQ8OAtV1/k0BsHI/4OD8+AICAwQF
BQYHBwcICQkJCQHPCQkJCQgHBwcGBQUEAwICXP4xAc9cAgIDBAUFBgcHBwgJCQkJ/jEJCQkJCAcHBwYFB
QQDAgICoHRYdD7e3oD9RAkJCAgIBwcGBgUEBAMCAQEBAQIDBAQFBgYHBwgICAkJzMwCvMzMCQkICAg
HBwYGBQQEAwIBAQEBAgMEBAUGBgcHCAgICQADAAAAAAOvA/QAAwBHAF0AAAERIREHERUfDTMhMz8
OES8OIyEjDw0nETMRITUhIw8NA1X+DFsCAgMEBQUGBgcICAgJCQkB9AkJCQgICAcGBgUFBAMCAQEBAQI
DBAUFBgYHCAgICQkJ/gwJCQkICAgHBgYFBQQDAgK2WQIT/e0JCQkIBwgHBgYFBAQDAgEC4/2EAnwF/YgJC
QgJCAcHBgYGBAQDAgICAgMEBAYGBgcHCAkICQkCeAkJCQgICAcGBgUFAwMDAQEDAwMFBQYGBwgICAk
Jsv2EAnxbAgIDBAUFBgYHCAgICQkAAAASAN4AAQAAAAAAAAABAAAAAQAAAAAAQAQAAEAAQAAAAA
AAgAHABEAAQAAAAAAAwAQABgAAQAAAAAABAAQACgAAQAAAAAABQALADgAAQAAAAAABgAQAEMA
AQAAAAAACgAsAFMAAQAAAAAACwASAH8AAwABBAkAAAACAJEAAwABBAkAAQAgAJMAAwABBAkAAg
AOALMAAwABBAkAAwAgAMEAAwABBAkABAAgAOEAAwABBAkABQAWAQEAAwABBAkABgAgARcAAwA
BBAkACgBYATcAAwABBAkACwAkAY8gdG9vbGJhci1tYXRlcmlhbFJlZ3VsYXJ0b29sYmFyLW1hdGVyaWFsdG
9vbGJhci1tYXRlcmlhbFZlcnNpb24gMS4wdG9vbGJhci1tYXRlcmlhbEZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5
bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAdABvAG8AbABiAGEAcgAtAG
0AYQB0AGUAcgBpAGEAbABSAGUAZwB1AGwAYQByAHQAbwBvAGwAYgBhAHIALQBtAGEAdABlAHIAaQB
hAGwAdABvAG8AbABiAGEAcgAtAG0AYQB0AGUAcgBpAGEAbABWAGUAcgBzAGkAbwBuACAAMQAuAD
AAdABvAG8AbABiAGEAcgAtAG0AYQB0AGUAcgBpAGEAbABGAG8AbgB0ACAAZwBlAG4AZQByAGEAdABl
AGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZ
ABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAAAAAAKAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAjAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4BDwEQAREBEgETA
RQBFQEWARcBGAEZARoBGwEcAR0BHgEfASABIQEiASMBJAAQVGV4dF9PdXRkZW50XzAwMQtQaWN0dX
JlXzAwMQxTZXR0aW5nc18wMDEQQ29sb3JfcGlja2VyXzAwMhBBbGlnbl9DZW50ZXJfMDA2CExpbmVfMD
AxDVVuZGVybGluZV8wMDEMU29ydF9aLUFfMDAxCFVuZG9fMDAxEENoYXJ0X2J1YmJsZV8wMDELRG93b
mxvYWRfMDAPVGV4dF9pbmRlbnRfMDAxEkNoYXJ0X0RvdWdobnV0XzAwMQlDbGVhcl8wMDINTnVtYm
VyaW5nXzAwMQxTb3J0X0EtWl8wMDEKSXRhbGljXzAwMQtCdWxsZXRzXzAwMQlQYXN0ZV8wMDEIUmV
kb18wMDEPQ2hhcnRfcmFkYXJfMDAxD0FsaWduX1JpZ2h0XzAwMQlUYWJsZV8wMDEOQWxpZ25fTGVm
dF8wMDEITWVudV8wMDEHQ3V0XzAwMghCb2xkXzAwMRFBbGlnbl9KdXN0aWZ5XzAwMQpSZWxvYWR
fMDAxClNlYXJjaF8wMDEKVXBsb2FkXzAwMQpEZXNpZ25fMDA1CkV4cG9ydF8wMDIQ29weV8wMDIAAA
==) format("truetype");

font-weight: normal;

font-style: normal;

}

.e-cut-icon:before {

```
content: "\e719";
}
.e-copy-icon:before {
content: "\e721";
}
.e-paste-icon:before {
content: "\e712";
}
.e-color-icon:before {
content: "\e703";
}
.e-bold-icon:before {
content: "\e71a";
}
.e-underline-icon:before {
content: "\e706";
}
.e-italic-icon:before {
content: "\e710";
}
.e-ascending-icon:before {
content: "\e70f";
}
.e-descending-icon:before {
content: "\e707";
}
.e-clear-icon:before {
content: "\e70d";
}
</style>
`
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-toolbar id="toolbar" width="800">
<e-items>
<e-item text="Cut" prefixIcon="e-cut-icon tb-icons"></e-item>
<e-item text="Copy" prefixIcon="e-copy-icon tb-icons"></e-item>
<e-item type="Separator"></e-item>
<e-item text="Bold" prefixIcon="e-bold-icon tb-icons"></e-item>
<e-item text="Underline" prefixIcon="e-underline-icon tb-icons"></e-item>
<e-item text="Italic" prefixIcon="e-italic-icon tb-icons"></e-item>
<e-item text="Color-Picker" prefixIcon="e-color-icon tb-icons"></e-item>
<e-item type="Separator"></e-item>
<e-item text="A-Z Sort" prefixIcon="e-ascending-icon tb-icons"></e-item>
<e-item text="Z-A Sort" prefixIcon="e-descending-icon tb-icons"></e-item>
<e-item text="Clear" prefixIcon="e-descending-icon tb-icons"></e-item>
</e-items>
</ejs-toolbar>
</template>
<script setup>
import { ToolbarComponent as EjsToolbar, ItemsDirective as EItems,
ItemDirective as EItem } from "@syncfusion/ej2-vue-navigations";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.e-tbar-btn .e-icons {
font-family: "Material_toolbar";
font-size: 16px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
}
.custom_bold .e-tbar-btn-text {
font-weight: 900;
}
.custom_italic .e-tbar-btn-text {
font-style: italic;
}
.custom_underline .e-tbar-btn-text {
text-decoration: underline red;
}
.e-txt-casing .e-tbar-btn-text {
font-variant: small-caps;
}
@font-face {
font-family: "Material_toolbar";
src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tShMAAAEoAAAAVmNtYXDoMOjqAAACDAAAAHhnbHlmIuy
19QAAAswAACNMaGVhZA6okZMAAADQAAAANmhoZWEIUQQkAAAArAAAACRobXR4jAAAAAAAAYAAAAC
MbG9jYYc0kUIAAAKEAAAASG1heHABOwG8AAABCAAAACBuYW1lx/RZbQAAJhgAAAKRcG9zdJZeEVU
AACisAAACGAABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAIwABAAAAAQAAAQsu/F8
PPPUACwQAAAAAANXLJlEAAAAA1csmUQAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAjAbAADgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnIQQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
```

```
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQ
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAA
AAAQAAAAEAAAABAAAAAQAAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAGQAAAAEAAQAAQAA5yH//wA
A5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcACAAJAAoACwAMAA0ADgAPABAAEQASABMAFAAVABY
AFwAYABkAGgAbABwAHQAeAB8AIAAhACIAAAAAADIAjgFwAfgCIAKYAxIDSAO2BRYFMAVcBnIGugb
2ByoHQgguCNYJRgn6CiQKiAquCsgMFgzADOYNzg7WDvAQyBEyEaYABwAAAAAD9APzAAMABwAKAA4
AEgAVABkAADchNSElITUhJTkBBSE1ITUhNSEFFxEnITUhDAPo/BgBtgIy/c7+SgG2AjL9zgIy/c7
+Svr6A+j8GAxefV67Pl19Xvr6AfScXgAAAAIAAAAAA/QD9AAEAEgAACUhNxc3AREfDyE/DxEvDyE
PDgOF/PbDisP9gQEBAwQEBgYICAgJCgoLCwsDCgsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQo
KCwsL/PYLCwsKCgkICAgGBgQEAwGz+qf6AYX89gsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQo
KCwsLAwoLCwsKCgkICAgGBgQEAwEBAQEDBAQGBggICAkKCgsLAAACAAAAAAPzA/QAQAC/AAABFQ8
PLw8/Dx8OAQ8ELwErAQ8FFR8FBxcPAxUfBzsBNx8LOwI/Cx8BOwE/Bj0BLwQ/Aic/BC8HKwEHLws
rAg8FArIBAgUGBwkKDAwODxAQERITEhIREQ8PDgwMCgkHBgUCAQECBQYHCQoMDA4PDxEREhITEhE
QEA8ODAwKCQcGBQL+zxUWFhUWfwUFBAUDBANqAgEBAgIDbgMDbwMCAQEBAmkDBAQEBQSEFBYWFxQ
CAgIDBAQEBcwFBAQEAwICAhQXFRYVgAQFBQQEAwNoAgEBAgIDcAEBAQNvAgIBAQEBA2gDBAQFBAW
DFBYWFxIBAgMDAwQFBcwFBAQDBAICAgAJCRIQEBAODgwLCgkHBgQDAQEDBAYHCQoLDA4OEBAQEhI
SEhAQEA4ODAsKCQcGBAMBAQMEBgcJCgsMDg4QEBASAc6ECwwNDjIBAQICA7QEBQQFBAMEUjIyVgM
EBAQFBAWwAwICATMODQwLhAQEBAMCAgECAgIDBAMEhAsMDQ4yAQECAgOwBAQFBAUEAwRSDAwaMlY
DBAQEBQQFsAMCAgEzDg0MC4QEAwQDAgICAgICAwQDAAAAAAMAAAAAA/MD2AAyADUAaQAAJRUfDTs
BPw41LwgPBwMhAScXAQ8GHQEfBQEfBjsBPwYBPwYvBwEDFgIDBAQGBgcICQkKCgsLCwsLCwoKCQg
ICAYGBAQDAQEDBAcMCQoLFCMtFQoJCQcFBHv96gEL04X+4gYFBAQCAgICAgIEBAUBNwcHBwgHCAc
IBwgHCAgHBwYBOAUEAwMCAQEBAQIDAwQFBv4PlwsLCwoKCQkIBwYGBAQDAgIDBAQGBgcICQkKCgs
LCwcPEBAYEBAPHCk3HRAQEBAQEAEIAQrThf7iBgcIBwgHCAgICAgHBwcH/skGBQQEAwIBAQIDBAQ
FBgE3BwcHBwgICAgHCAgHCAcGAfEABQAAAAAD9APzAAMABwALAA8AEwAANyE1ITchNSEnITUhNyE
1ISchNSEMA+j8GN4CLP3U3gPo/BjeAiz91N4D6PwYDF6AW5xefVqAXgAAAAAEAAAAAAP0A/QACQA
TABcAWwAAAQcVMzcXMzUjLwEjFTMbATM1IwElESERBxEfDyE/DxEvDyEPDgFro8ObnnROxOp0nZv
qTij+8AGW/NReAQEDBAQGBggICAkKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJCgo
LCwv89gsLCwoKCQgICAYGBAQDAQENAyOWliO4BSUBK/7VJQFSffzUAywR/PYLCwsKCgkICAgGBgQ
EAwEBAQEDBAQGBggICAkKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJCgoLCwAAAAA
CAAAAAAOWA/QAAwBpAAA3ITUhExUfHTsBPx01ESMRDw8vDxEjagMs/NRKAgIDAwUFBgcHCAkJCgs
LCwwNDQ0ODw4PEA8QERARERERERBEQDxAPDg8ODQ0NDAsLCwoJCQgHBwYFBQMDAgKLAQMFBggKCww
ODxARERMTFBQTExEREA8ODAsFCQcGBAKLDH0BsBERERERQEA8QDg8ODg0MDQsLCwoJCQgHBwYFBQQ
DAgEBAgMEBQUGBggICQkKCgsMDAwNDg4ODw8PEBAQERERAb7+RRQTEhIREA8NDQsKCAYFAwEBAwU
GCAoLDQ0PCBASEhMTAcUABQAAAAAD9APXAAIABQANABcAGgAAJTcjASM3ATM3MxczAyMFIQEVITU
hATUhJTMnAgJx4wG/vl/+/Fot+i1a3FD9RgEg/t4Bov7UAST+aAF/6XQobQET//47eHgCM07+XD5
NAaQ/UHMAAAAAAQAAAAAD9ALoAF8AABMhJz8PHxo3Lx8PDycMAbWyDQ0ODg8PDxAQEBERERIREhA
QEBAQDw8PDw4ODg0NDQwMFxYTEhAHBgYGBXUHBwgJCQoLCwwNDQ0PDg8QEBERERITEhMUExQVFBU
VFRgYFxcXFhYVFhQUFBMTEhGwARi6CwsJCggICAYGBgQEAwIBAQEBAgIDBAQFBQYHBwgICQkKFRY
YGhsODg8PDygUFBMTEhISERARDw8PDg0NDAsLCgoICAgGBgQEAwMBAQECAwQFBgcJCQoLDA0ODg+
6AAYAAAAAA/MD9AA/AGsAqwDrAO8BMwAAARUfDTsBPw09AS8ODw4lHwk7AT8IPQEvByMnByMPByU
fDz8PLw8PDiUfDz8OPQEvDSsBDw0lESERBxEfDyE/DxEvDyEPDgHhAgMFBQYHCAkKCgsLDA0NDA0
MCwsKCgkIBwYFBQMCAgMFBQYHCAkKCgsLDA0MDQ0MCwsKCgkIBwYFBQMC/scBAQEFBwgKCwYGBwY
GBgwKCAcFAQEBAQUHCAoMBgYGBwYGCwoIBwUBAQHzAQECBAQEBgYGCAcICQkJCgoJCQgJBwgGBgY
EBAMDAQEBAQMDBAQGBgYIBwkICQkKCgkJCQgHCAYGBgQEBAIB/qgBAQMEBAYGBwgICQoKCgsLCws
LCgkJCQcHBwUFAwMCAgMDBQUHBwcJCQkKCwsLCwsKCgoJCAgHBgcHCAkJCQkAyYKCQkJCQgHBwYG
BQQEAgEBAQECBAQFBgYHBwgJCQkJCvzaCgkJCQkIBwcGBgUEAwMBAWQNDAwMCwoKCQgHBgUFAwICAwUFBgcICQoKCwwMDA0NDAwLCwsJCQgHBwUEAwIBAQIDBAUHBwgJCQsLCwwMMQYGBgsKCQcEAgEBAgQHCQoLBgYGBwYGCwoJBgUCAQECBQYJCgsGBvMJCgkICAgHBwcICQkJCQoDJgoJCQkJCAcHBwUDAwICAwMFBQcHBwkJCQsLCwsKCQkJBwcHBwYFBQQDAwEBAQIDBAUFBwcICQkLCwsKCQkJBwcHBwUFBAMDAQEBAgMEBQUHBwgJCQsLCwoJCQkHBwcHBQUEAwMBAQECAwQFBQcHCAkJCwsLCgkJCQcHBwc
```

```
ICgoL9g4OHA4ODQ4NDg0NDQwNDKkLDA8HCAkJCAgICA4DEAUFBQMEAu4EBwkKDQ8REwoKCwsMDAw
NDQ4ODg4PDy8KAZP81F4BAQMDBAUGBgcHCAkJCQkKAyYKCQkJCQgHBwYGBQQEAgEBAQECBAQFBgY
HBwgJCQkJCvzaCgkJCQkIBwcGBgUEAwMBAZz0AgMDBAQFBQYHBwgICQkJCgsLCwsYGhsbDw4PDwo
KCQgIBwaUDAsLCgkIBwcGBQQEAgIBAfEBAwMEBQYIBQcGBw8PEA8ODa0NDQ0ODg4ODw8PDw8PEA8
PEA8PEA8ODw8ODg0ODQ0MAj8BBAMDBAQFBgcHBwkJCqoGBQQBAQICBAMJEAcHCAgJCh0dHRsaGRc
XCgoKCQgICAcGBgYEBQMDBj781AMsA/zaCgkJCQkIBwcGBgUEAwMBAQEBAgQEBQYGBwcICQkJCQo
DJgoJCQkJCAcHBgYFBAQCAQEBAQIEBAUFBwcHCAkJCQkAAgAAAAAD8wPrAB8AMwAAEw8HHww/BBU
hNSEBNwkBPwcvCDYKCAcGBQMCAQECAwUGBwgKrwgJCgoKCgkINQKQ/Xv+20EBPAGOCgkHBgUDAgE
BAgMFBgcJCtMBoQsMDQ0NDg4ODg0ODQ0NDAuvBgUDAQECBAY0I14BJEH+xAGRCwwMDQ0ODg4ODg4
NDQwMC9QABgAAAAAD9AP0AAMADwATAB0AIQAnAAAlITUhIzMVIxUzFSMVMzUjNyE1ISMzBxUzNSM
3NSM3ITUhJzMVMzUjAQYC7v0S+n0/P328vPoC7v0S+nh4vHh4vPoC7v0S+j4/fWpeID4gPvrbXXw
/P3w/u14gvPoAAAAABQAAAAAD9APbAAIABQANABcAGgAAJTcjAyM3ATM3MxczAyMFIQEVITUhATU
hJzMnAgJx4x6+Xv76Wi39LF3fTwFlAST+3AGk/tIBJP5mw+l0JXMBGP/+N3h4AjRQ/lo+TQGpPk5
zAAABAAAAAAOsA/QACwAAATMDIxUhNSMTMzUhAXGd88gCPJ3zyP3EAx79xNbWAjzWAAAGAAAAAAP
0A9QAAwBDAEcAhwCLAMsAACUhNSEHFR8OPw49AS8ODw4TITUhBxUfDTsBPw09AS8ODw4TITUhBxU
fDj8OPQEvDg8OAQYC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgYGBQQEAwMCAQECAwMEBAUGBgY
HBwgICAgIBwgGBwYFBQUEAwMBAvoC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgYGBQQEAwMCAQE
CAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAvoC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgY
GBQQEAwMCAQECAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAkpeLwgIBwcHBwYFBQUDBAICAQE
BAQICBAMFBQUGBwcHBwgICAgIBwcGBgYFBAQDAwIBAQEBAgMDBAQFBgYGBwcICAFgXS4ICAgHBwY
GBgUEBAMDAgEBAgMDBAQFBgYGBwcICAgICAcHBwcGBQUFAwQCAgEBAQECAgQDBQUFBgcHBwcIAUB
dLggICAcHBgYGBQQEAwMCAQEBAQIDAwQEBQYGBgcHCAgICAgHBwcHBgUFBQMEAgIBAQEBAgIEAwU
FBQYHBwcHCAAAAwAAAAADmQP0AAcAKACNAAABFSE1MxEhESUHFQ8GLwc/Bx8GJysBDw0VERUfDTM
hMz8NNRE1Lw0rAS8OKwEPDQEdAcZb/YQBbAEDBAYHBwkJCQkHBwYEAwEBAwQGBwcJCQkJBwcGBAO
svwkJCQgICAcGBgYEBAMCAgICAwQEBgYGBwgICAkJCQJ8CQkJCAgIBwYGBgQEAwICAgIDBAQGBgY
HCAgICQkJvwMFBQYGBwgICQkJCgoKCwsLCwoKCgkJCQgIBwYGBQUDPoiI/SkC1y4FBQgIBwUEAwE
BAwQFBwgICgkICAcFBQIBAQIFBQcICCQCAgMEBAYGBgcICAgJCQn9KQkJCQgICAcGBgUFBAMCAgI
CAwQFBQYGBwgICAkJCQLXCQkJCAgIBwYGBgQEAwICCgkJCAgIBwYGBQQEAwICAgIDBAQFBgYHCAg
ICQkAAQAAAAAD9ALoAGAAAAAExLw8PHxc/Gh8PByERA0QREhMTFBQUFRYWFhcXFxgYFRUVFBUUExQ
TEhMSEREREBAPDg8NDQ0MCwsKCQkIBwd1BQYGBgcQEhMWFwwMDQ0NDg4ODw8PDxAQEBAQEhESERE
QERAQDw8PDg4NDbIBtQIuDw4ODQwLCgkJBwYFBAMCAQEBAwMEBAYGCAgICgoLCwwNDQ4PDw8REBE
SEhITExQUKA8PDw4OGxoYFhUKCQkICAcHBgUFBAQDAgIBAQEBAgMEBAYGBggICAoJCwu6AdAAAAA
OAAAAAAP0A/MAAgAFAAgACwAQABQAFwAbAB4AIQApAC0AMQB1AAABETclFzUXNyMFNyETFQUhEQE
hJRMlMycFMSEnBzcnBxcRBRMDBSUDEy0BEQMlIwUDEQcRHw8hPw8RLw8hDw4CGcj+ZaG3MJb+wM7
+4jQBCv6EAy7+ggEKdP1S3JkBCwEjWemWlvrIATJ0dP7n/up3dwEWAZhy/vQ0/vZyXgEBAwQEBgY
ICAgJCgoLCwsDCgsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQoKCwsL/PYLCwsKCgkICAgGBgQ
EAwEBxv7fWSQ730Bky8v+9QNxAYH+f28BHx2ZmcukmTgJywEeP/7n/ud3dwEZAR13Bv5xAR1ycv7
yAYAR/PYLCwsKCgkICAcHBQUEAwEBAQEDBAUFBwcICAkKCgsLCwMKCwsLCgoJCAgHBwUFBAMBAQE
BAwQEBgYIBwkJCgoLCwAAAAAFAAAAAAP0A/MAAwAHAAsADwATAAA3ITUhJSE1ISUhNSElITUhJSE
1IQwD6PwYAVgCkP1w/qgD6PwYAVgCkP1w/qgD6PwYDF6AW5xefV19XgAAAAAKAAAAAAP0A/MAAwA
HAAsADwATABcAGwAfACMARwAAARUjNSMVIzUjFSM1ARUjNSMVIzUjFSM1JRUjNSMVIzUjFSM1JxE
fByE/BxEvByEPBgOW+j7bP9oDLPo+2z/aAyz6Pts/2l4BAwUGAwgJCgOJCgkJBwYDBAIBAwUGAwg
JCvx3CgkJBwYFAwElvb27u7u7ARrb29vb29v6vLy8vLy8hvyCCwoJBwQGBAIBAwUHBwUJCgOECwo
JBwQGBAIBAwUGCAkKAAAAAAUAAAAA/QD8wADchNSE1IwVHw8zPDy8PPw8fEAE1IwUVHw8zPwMXBy8FDw8fDz8PNS8NNwEzNQE
/BS8PDw4BOAEBAwMEBQYGBwgICQkKCgoJCQgIBwYGBQUEAwMBAQEDAwQFBgYHCAgJCQoKCgkJCAgHBgYFBQQDAwIC4QEBAwMEBQYGBwgICAkJCgoKCgkJCAgHBgYFBQQDAwIC4QEBAwMEBQYGBwgICAkJCgoKCgkJCAgHBgYFBQQDAeICAgMDBQUFBQUFAwMCAgICAwMFBQUFBQUDAwIC4QEBAwMEBQYGBwgICAkJCgoKCgkJCAgHBgYFBQQDAftkAV6W/K4
BAwUHCAoMDQ4PERETExQUCwsMDRESExQUFBQTExERDw4NDAoIBwUEAwECAwUHCAoMDQ4PERET
```

```
DBQcICgwNDg8RERMTFAADAAAAAANXA7UAIgBFAJMAAAEzHw4PDisBNRMzHw4PDisBNQMhPxEvDz8
PLxghAkgKCgkJCAgHBwYGBAQEAgEBAQEDAwQFBgYHBwgJCAkKCeDACgoJCQgIBwcGBgQEBAIBAQE
BAgQEBAYGBwcICAkJCgrAwAHDDQwMDBcWFRMSEQ8NDAoHBgQBAQIDBAYHBwkKCgsNDA4ODwsLCgo
KCAgIBgYFBQMDAQEBAQECAwQEBAUGDA8QEhQVFgwMDA0NDQ0N/nABogICAwQEBgYGBwgICQkKCQo
KCQgJBwgGBgUFBAMCArsBdwICAwQEBgYGCAcICQkKCQoKCQkIBwgGBgYEBAMCArv9MQEBAQIGCAo
MDg8REhQUFhcYGBERERAQEA4ODgwMDAoJCQcICQkKCgoLDAsMDAwMDQwNDQwNDQwMCwwLCxQUERA
ODQoFAwQDAgEBAQAABQAAAAAD9APzAAMABwALAA8AEwAANyE1ITUhNSE1ITUhNSE1ITUhNSEMA+j
8GAPo/BgD6PwYA+j8GAPo/BgMXn1enF59XX1eAAAAAAEAAAAAA9QD1ADUAAATHx8/DxcRIRcPDy8
fPx8fDzMvHw8eKwECAwQFBggICQoMDA0ODhAQERISExQUFRUWFhcXGBgYGBgXFxcWFhUVFBQTEhI
REIr+ZrsMDA0ODg4PEBAQEBESERISEhIREhEQEQ8QDw8ODg0NDAwLCgoJCQgHBgYEBAQCAQEBAQI
EBAQGBgcICQkKCgsMDA0NDg4PDxAPERAREhESEhwcGxoaGBgWFRQSEQ8OCwp7BQYHCAgJCQoLCww
NDQ4ODg8QEBEREriSEhMTFBMUFRQYGBgXFxYWFRUUFBMSEhEQEA4ODQwMCgkICAYFBAMCAgAYGBc
XFxYWFRUUFBMSEhEQEA4ODQ0LCgoICAYFBAMCAQECAwQFBggICgoLDQ0ODhCKAZq7DAsLCgkJCAc
HBQUEAwMBAQEBAgQEBAYGBwgICgkLCwwMDQ0ODg8PDxAREBESERISEhIREhEQERAPDw8ODg0NDAw
LCwkKCAgHBgYEBAQCAQECAwUICQsNDxASExUWFxgaExITERIREBAQDw8ODg0NDAsLCgoJCAcHBgY
EBAMCAQEBAgMEBQYICAoKCw0NDg4QEBESEhMUFBUVFhYXFxcYAAAAAgAAAAD8gP0AGcA7gAAARU
PGC8YPQE/FzsBHxcFHx8/DxcVATcBIyc/Dj0BLx0rAQ8dAoABAgIDAwQFBQUNDxATExYLCwwMDAw
NDQ0NDQ0NDQwNDAsMCxUUEhAPDQUFBQQDAwMBAQEBAwMDBAUFBQ0PEBIUFQsMCwwNDA0NDQ0NDQ0
NDAwMDAsLFhMTEA8NBQUFBAMDAgIB/Y0BAQMDBAUGBggICQkLCwsNDA4ODg8QEBARERISEhMTExE
REBEQEBAQDw8ODg4ODA0OAR1W/uMuDgoKCQkIBwYGBgQEAwMCAQICAwQFBgcHCAkKCgsMDA0NDg8
PDxARERERER
```

```
AYQB0AGUAcgBpAGEAbABSAGUAZwB1AGwAYQByAHQAbwBvAGwAYgBhAHIALQBtAGEAdABlAHIAaQB
hAGwAdABvAG8AbABiAGEAcgAtAG0AYQB0AGUAcgBpAGEAbABWAGUAcgBzAGkAbwBuACAAMQAuADA
AdABvAG8AbABiAGEAcgAtAG0AYQB0AGUAcgBpAGEAbABGAG8AbgB0ACAAZwBlAG4AZQByAGEAdAB
lAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHU
AZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAAAAAAKAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAjAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4BDwEQAREBEgE
TARQBFQEWARcBGAEZARoBGwEcAR0BHgEfASABIQEiASMBJAAQVGV4dF9PdXRkZW50XzAwMQtQaWN
0dXJlXzAwMQxTZXR0aW5nc18wMDEQQ29sb3JfcGlja2VyXzAwMhBBbGlnbl9DZW50ZXJfMDA2CEx
pbmVfMDAxDVVuZGVybGluZV8wMDEMU29ydF9aLUFfMDAxCFVuZG9fMDAxEENoYXJ0X2J1YmJsZV8
wMDELRG93bmxvYWRfMDAPVGV4dF9pbmRlbnRfMDAxEkNoYXJ0X0RvdWdobnV0XzAwMQlDbGVhc18
wMDINTnVtYmVyaW5nXzAwMQxTb3J0X0EtWl8wMDEKSXRhbGljXzAwMQtCdWxsZXRzXzAwMQlQYXN
0ZV8wMDEIUmVkb18wMDEPQ2hhcnRfcmFkYXJfMDAxD0FsaWduX1JpZ2h0XzAwMQlUYWJsZV8wMDE
OQWxpZ25fTGVmdF8wMDEITWVudV8wMDEHQ3V0XzAwMghCb2xkXzAwMRFBbGlnbl9KdXN0aWZ5XzA
wMQpSZWxvYWRfMDAxClNlYXJjaF8wMDEKVXBsb2FkXzAwMQpEZXNpZ25fMDA1CkV4cG9ydF8wMDE
IQ29weV8wMDIAAA==) format("truetype");
font-weight: normal;
font-style: normal;
}
.e-cut-icon:before {
content: "\e719";
}
.e-copy-icon:before {
content: "\e721";
}
.e-paste-icon:before {
content: "\e712";
}
.e-color-icon:before {
content: "\e703";
}
.e-bold-icon:before {
content: "\e71a";
}
.e-underline-icon:before {
content: "\e706";
}
.e-italic-icon:before {
content: "\e710";
}
.e-ascending-icon:before {
content: "\e70f";
}
.e-descending-icon:before {
content: "\e707";
}
.e-clear-icon:before {
content: "\e70d";
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-toolbar id="toolbar" width="800">
<e-items>
<e-item text="Cut" prefixIcon="e-cut-icon tb-icons"></e-item>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-item text="Copy" prefixIcon="e-copy-icon tb-icons"></e-item>
<e-item type="Separator"></e-item>
<e-item text="Bold" prefixIcon="e-bold-icon tb-icons"></e-item>
<e-item text="Underline" prefixIcon="e-underline-icon tb-icons"></e-item>
<e-item text="Italic" prefixIcon="e-italic-icon tb-icons"></e-item>
<e-item text="Color-Picker" prefixIcon="e-color-icon tb-icons"></e-item>
<e-item type="Separator"></e-item>
<e-item text="A-Z Sort" prefixIcon="e-ascending-icon tb-icons"></e-item>
<e-item text="Z-A Sort" prefixIcon="e-descending-icon tb-icons"></e-item>
<e-item text="Clear" prefixIcon="e-descending-icon tb-icons"></e-item>
</e-items>
</ejs-toolbar>
</template>
<script>
import {
ToolbarComponent,
ItemDirective,
ItemsDirective,
} from "@syncfusion/ej2-vue-navigations";
export default {
name: "App",
components: {
"ejs-toolbar": ToolbarComponent,
"e-items": ItemsDirective,
"e-item": ItemDirective
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.e-tbar-btn .e-icons {
font-family: "Material_toolbar";
font-size: 16px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
}
.custom_bold .e-tbar-btn-text {
font-weight: 900;
}
.custom_italic .e-tbar-btn-text {
font-style: italic;
}
.custom_underline .e-tbar-btn-text {
text-decoration: underline red;
}
.e-txt-casing .e-tbar-btn-text {
font-variant: small-caps;
}
@font-face {
font-family: "Material_toolbar";
```

```
src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tShMAAAEoAAAAVmNtYXDoMOjqAAACDAAAAHhnbHlmIuy
19QAAAswAACNMaGVhZA6okZMAAADQAAAANmhoZWEIUQQkAAAArAAAACRobXR4jAAAAAAAAYAAAAC
MbG9jYYc0kUIAAAKEAAAASG1heHABOwG8AAABCAAAACBuYW1lx/RZbQAAJhgAAAKRcG9zdJZeEVU
AACisAAACGAABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAIwABAAAAAQAAAQsu/F8
PPPUACwQAAAAAANXLJlEAAAAA1csmUQAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAjAbAADgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnIQQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQ
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAA
AAAQAAAAEAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAGQAAAAEAAQAAQAA5yH//wA
A5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcACAAJAAoACwAMAA0ADgAPABAAEQASABMAFAAVABY
AFwAYABkAGgAbABwAHQAeAB8AIAAhACIAAAAAADIAjgFwAfgCIAKYAxIDSAO2BRYFMAVcBnIGugb
2ByoHQgguCNYJRgn6CiQKiAquCsgMFgzADOYNzg7WDvAQyBEyEaYABwAAAAAD9APzAAMABwAKAA4
AEgAVABkAADchNSElITUhJTkBBSE1ITUhNSEFFxEnITUhDAPo/BgBtgIy/c7+SgG2AjL9zgIy/c7
+Svr6A+j8GAxefV67Pl19Xvr6AfScXgAAAAIAAAAAA/QD9AAEAEgAACUhNxc3AREfDyE/DxEvDyE
PDgOF/PbDisP9gQEBAwQEBgYICAgJCgoLCwsDCgsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQo
KCwsL/PYLCwsKCgkICAgGBgQEAwGz+qf6AYX89gsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQo
KCwsLAwoLCwsKCgkICAgGBgQEAwEBAQEDBAQGBggICAkKCgsLAAACAAAAAAPzA/QAQAC/AAABFQ8
PLw8/Dx8OAQ8ELwErAQ8FFR8FBxcPAxUfBzsBNx8LOwI/Cx8BOwE/Bj0BLwQ/Aic/BC8HKwEHLws
rAg8FArIBAgUGBwkKDAwODxAQERITEhIREQ8PDgwMCgkHBgUCAQECBQYHCQoMDA4PDxEREhITEhE
QEA8ODAwKCQcGBQL+zxUWFhUWfwUFBAUDBANqAgEBAgIDbgMDbwMCAQEBAmkDBAQEBQSEFBYWFxQ
CAgIDBAQEBcwFBAQEAwICAhQXFRYVgAQFBQQEAwNoAgEBAgIDcAEBAQNvAgIBAQEBA2gDBAQFBAW
DFBYWFxIBAgMDAwQFBcwFBAQDBAICAgAJCRIQEBAODgwLCgkHBgQDAQEDBAYHCQoLDA4OEBAQEhI
SEhAQEA4ODAsKCQcGBAMBAQMEBgcJCgsMDg4QEBASAc6ECwwNDjIBAQICA7QEBQQFBAMEUjIyVgM
EBAQFBAWwAwICATMODQwLhAQEBAMCAgECAgIDBAMEhAsMDQ4yAQECAgOwBAQFBAUEAwRSDAwaMlY
DBAQEBQQFsAMCAgEzDg0MC4QEAwQDAgICAgICAwQDAAAAAAMAAAAAA/MD2AAyADUAaQAAJRUfDTs
BPw41LwgPBwMhAScXAQ8GHQEfBQEfBjsBPwYBPwYvBwEDFgIDBAQGBgcICQkKCgsLCwsLCwoKCQg
ICAYGBAQDAQEDBAcMCQoLFCMtFQoJCQcFBHv96gEL04X+4gYFBAQCAgICAgIEBAUBNwcHBwgHCAc
IBwgHCAgHBwYBOAUEAwMCAQEBAQIDAwQFBv4PlwsLCwoKCQkIBwYGBAQDAgIDBAQGBgcICQkKCgs
LCwcPEBAYEBAPHCk3HRAQEBAQEAEIAQrThf7iBgcIBwgHCAgICAgHBwcH/skGBQQEAwIBAQIDBAQ
FBgE3BwcHBwgICAgHCAgHCAcGAfEABQAAAAAD9APzAAMABwALAA8AEwAANyE1ITchNSEnITUhNyE
1ISchNSEMA+j8GN4CLP3U3gPo/BjeAiz91N4D6PwYDF6AW5xefVqAXgAAAAAEAAAAAAP0A/QACQA
TABcAWwAAAQcVMzcXMzUjLwEjFTMbATM1IwElESERBxEfDyE/DxEvDyEPDgFro8ObnnROxOp0nZv
qTij+8AGW/NReAQEDBAQGBggICAkKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJCgo
LCwv89gsLCwoKCQgICAYGBAQDAQENAyOWliO4BSUBK/7VJQFSffzUAywR/PYLCwsKCgkICAgGBgQ
EAwEBAQEDBAQGBggICAkKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJCgoLCwAAAAA
CAAAAAAOWA/QAAwBpAAA3ITUhExUfHTsBPx01ESMRDw8vDxEjagMs/NRKAgIDAwUFBgcHCAkJCgs
LCwwNDQ0ODw4PEA8QERARERERERBEQDxAPDg8ODQ0NDAsLCwoJCQgHBwYFBQMDAgKLAQMFBggKCww
ODxARERMTFBQTExEREA8ODAsFCQcGBAKLDH0BsBERERERQEA8QDg8ODg0MDQsLCwoJCQgHBwYFBQQ
DAgEBAgMEBQUGBggICQkKCgsMDAwNDg4ODw8PEBAQERERAb7+RRQTEhIREA8NDQsKCAYFAwEBAwU
GCAoLDQ0PCBASEhMTAcUABQAAAAAD9APXAAIABQANABcAGgAAJTcjASM3ATM3MxczAyMFIQEVITU
hATUhJTMnAgJx4wG/vl/+/Fot+i1a3FD9RgEg/t4Bov7UAST+aAF/6XQobQET//47eHgCM07+XD5
NAaQ/UHMAAAAAAQAAAAAD9ALoAF8AABMhJz8PHxo3Lx8PDycMAbWyDQ0ODg8PDxAQEBERERIREhA
QEBAQDw8PDw4ODg0NDQwMFxYTEhAHBgYGBXUHBwgJCQoLCwwNDQ0PDg8QEBERERITEhMUExQVFBU
VFRgYFxcXFhYVFhQUFBMTEhGwARi6CwsJCggICAYGBgQEAwIBAQEBAgIDBAQFBQYHBwgICQkKFRY
YGhsODg8PDygUFBMTEhISERARDw8PDg0NDAsLCgoICAgGBgQEAwMBAQECAwQFBgcJCQoLDA0ODg+
6AAYAAAAAA/MD9AA/AGsAqwDrAO8BMwAAARUfDTsBPw09AS8ODw4lHwk7AT8IPQEvByMnByMPBYU
fDz8PLw8PDiUfDz8OPQEvDSsBDw0lESERBxEfDyE/DxEvDyEPDgHhAgMFBQYHCAkKCgsLDA0NDA0
MCwsKCgkIBwYFBQMCAgMFBQYHCAkKCgsLDA0MDQ0MCwsKCgkIBwYFBQMC/scBAQEFBwgKCwYGBwY
GBgwKCAcFAQEBAQUHCAoMBgYGBwYGCwoIBwUBAQHzAQECBAQEBgYGCAcICQkJCgoJCQgJBwgGBgY
EBAMDAQEBAQMDBAQGBgYIBwkICQkKCgkJCQgHCAYGBgQEBAIB/qgBAQMEBAYGBwgICQoKCgsLCws
LCgkJCQcHBwUFAwMCAgMDBQUHBwcJCQkKCwsLCwsKCgoJCAgHBgYEBAMBAlD81F4BAQMDBAUGBgc
HCAkJCQkKAyYKCQkJCQgHBwYGBQQEAgEBAQECBAQFBgYHBwgJCQkJCvzaCgkJCQkIBwcGBgUEAwM
BAWQNDAwMCwoKCQgHBgUFAwICAwUFBgcICQoKCwwMDA0NDAwLCwsJCQgHBwUEAwIBAQIDBAUHBwg
JCQsLCwwMMQYGBgsKCQcEAgEBAgQHCQoLBgYGBwYGCwoJBgUCAQECBQYJCgsGBvMJCgkICAgHBwY
FBQQDAwEBAQEDAwQFBQYHBwgICAkKCQoJCQkICAcGBwUFBAMCAQEBAQIDBAUFBwYHCAgICQkKCws
```

KCgoJCAgHBgYEBAMBAQEBAwQEBgYHCAgJCgoKCwsLCwsKCQkJBwcHBQUDAwICAwMFBQcHBwkJCQo
LC9/81AMsA/zaCgkJCQkIBwcGBgUEBAIBAQEBAgQEBQUHBwcICQkJCQoDJgoJCQkJCAcHBwUFBAQ
CAQEBAQIEBAUFBwcHCAkJCQkAAAAAAgAAAAADtQP0AAMACgAANyE1IQEjCQEjESFKA2z8lAEG7AG
cAZzs/qAMfQIK/mQBnAFhAAYAAAAAA/QD8wADAAcACwAPABIAFgAANyE1ISUhNSE1ITUhNSE1KQE
RNwMhNSEMA+j8GAG2AjL9zgIy/c4CMv3O/kr6+gPo/BgMXn1efV19Xv4M+gGWXgAFAAAAAAPzA/M
AJQBpAKgArADwAAABFT8bIw8GBR8PNSMvDT8CJw8OHw8TByMPDBc/AzsBHwUzHwYzLxcjJREhEQc
RHw8hPw8RLw8hDw4CKg8QDw8ODg4NDQwMDAwKCwoKCQgJDw0KCQQDAgLxBAUGBggJC/7WDQ0NDg4
PDw8PEA8QEBAQEAoKCQkJCQgIBgQEBAUDAQEDBKsKCQgIBwYGBQUDAwMBAQEBAQEDAwQEBQYHBwg
ICgoL9g4OHA4ODQ4NDg0NDQwNDKkLDA8HCAkJCAgICA4DEAUFBQMEAu4EBwkKDQ8REwoKCwsMDAw
NDQ4ODg4PDy8KAZP81F4BAQMDBAUGBgcHCAkJCQkKAyYKCQkJCQgHBwYGBQQEAgEBAQECBAQFBgY
HBwgJCQkJCvzaCgkJCQkIBwcGBgUEAwMBAZz0AgMDBAQFBQYHBwgICQkJCgsLCwsYGhsbDw4PDwo
KCQgIBwaUDAsLCgkIBwcGBQQEAgIBAfEBAwMEBQYIBQcGBw8PEA8ODa0NDQ0ODg4ODw8PDw8PEA8
PEA8PEA8ODw8ODg0ODQ0MAj8BBAMDBAQFBgcHBwkJCqoGBQQBAQICBAMJEAcHCAgJCh0dHRsaGRc
XCgoKCQgICAcGBgYEBQMDBj781AMsA/zaCgkJCQkIBwcGBgUEAwMBAQEBAgQEBQYGBwcICQkJCQo
DJgoJCQkJCAcHBgYFBAQCAQEBAQIEBAUFBwcHCAkJCQkAAgAAAAAD8wPrAB8AMwAAEw8HHww/BBU
hNSEBNwkBPwcvCDYKCAcGBQMCAQECAwUGBwgKrwgJCgoKCgkINQKQ/Xv+20EBPAGOCgkHBgUDAgE
BAgMFBgcJCtMBoQsMDQ0NDg4ODg0ODQ0NDAuvBgUDAQECBAY0I14BJEH+xAGRCwwMDQ0ODg4ODg4
NDQwMC9QABgAAAAAD9AP0AAMADwATAB0AIQAnAAAlITUhIzMVIxUzFSMVMzUjNyE1ISMzBxUzNSM
3NSM3ITUhJzMVMzUjAQYC7v0S+n0/P328vPoC7v0S+nh4vHh4vPoC7v0S+j4/fWpeID4gPvrbXXw
/P3w/u14gvPoAAAAABQAAAAAD9APbAAIABQANABcAGgAAJTcjAyM3ATM3MxczAyMFIQEVITUhATU
hJzMnAgJx4x6+Xv76Wi39LF3fTwFlAST+3AGk/tIBJP5mw+l0JXMBGP/+N3h4AjRQ/lo+TQGpPk5
zAAABAAAAAAOsA/QACwAAATMDIxUhNSMTMzUhAXGd88gCPJ3zyP3EAx79xNbWAjzWAAAGAAAAAAP
0A9QAAwBDAEcAhwCLAMsAACUhNSEHFR8OPw49AS8ODw4TITUhBxUfDTsBPw09AS8ODw4TITUhBxU
fDj8OPQEvDg8OAQYC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgYGBQQEAwMCAQECAwMEBAUGBgY
HBwgICAgIBwgGBwYFBQUEAwMBAvoC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgYGBQQEAwMCAQE
CAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAvoC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgY
GBQQEAwMCAQECAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAkpeLwgIBwcHBwYFBQUDBAICAQE
BAQICBAMFBQUGBwcHBwgICAgIBwcGBgYFBAQDAwIBAQEBAgMDBAQFBgYGBwcICAFgXS4ICAgHBwY
GBgUEBAMDAgEBAgMDBAQFBgYGBwcICAgICAcHBwcGBQUFAwQCAgEBAQECAgQDBQUFBgcHBwcIAUB
dLggICAcHBgYGBQQEAwMCAQEBAQIDAwQEBQYGBgcHCAgICAgHBwcHBgUFBQMEAgIBAQEBAgIEAwU
FBQYHBwcHCAAAAwAAAAADmQP0AAcAKACNAAABFSE1MxEhESUHFQ8GLwc/Bx8GJysBDw0VERUfDTM
hMz8NNRE1Lw0rAS8OKwEPDQEdAcZb/YQBbAEDBAYHBwkJCQkHBwYEAwEBAwQGBwcJCQkJBwcGBAO
svwkJCQgICAcGBgYEBAMCAgICAwQEBgYGBwgICAkJCQJ8CQkJCAgIBwYGBgQEAwICAgIDBAQGBgY
HCAgICQkJvwMFBQYGBwgICQkJCgoKCwsLCwoKCgkJCQgIBwYGBQUDPoiI/SkC1y4FBQgIBwUEAwE
BAwQFBwgICgkICAcFBQIBAQIFBQcICCQCAgMEBAYGBgcICAgJCQn9KQkJCQgICAcGBgUFBAMCAgI
CAwQFBQYGBwgICAkJCQLXCQkJCAgIBwYGBgQEAwICCgkJCAgIBwYGBQQEAwICAgIDBAQFBgYHCAg
ICQkAAQAAAAAD9ALoAGAAAAExLw8PHxc/Gh8PByERA0QREhMTFBQUFRYWFhcXFxgYFRUVFBUUExQ
TEhMSERERERBAPDg8NDQ0MCwsKCQkIBwd1BQYGBgcQEhMWFwwMDQ0NDg4ODw8PDxAQERAQDw8PDg4NDbIBtQIuDw4ODQwLCgkJBwYFBAMCAQEBAwMEBAYGCAgICgoLCwwNDQ4PDw8REBE
SEhITExQUKA8PDw4OGxoYFhUKCQkICAcHBgUFBAQDAgIBAQEBAgMEBAYGBggICAoJCwu6AdAAAAA
OAAAAAAP0A/MAAgAFAAgACwAQABQAFwAbAB4AIQApAC0AMQB1AAABETclFzUXNyMFNyETFQUhEQE
hJRMlMycFMSEnBzcnBxcRBRMDBSUDEy0BEQMlIwUDEQcRHw8hPw8RLw8hDw4CGcj+ZaG3MJb+wM7
+4jQBCv6EAy7+ggEKdP1S3JkBCwEjWemWlvrIATJ0dP7n/up3dwEWAZhy/vQ0/vZyXgEBAwQEBgY
ICAgJCgoLCwsDCgsLCwoKCQgICAYGBwgICAkJCQLXCQkJCAgIBwYGBgQEAwICCgkJCAgIBwYGBQQEAwICAgIDBAQFBgYHCAgICQkAAQAAAAAD9ALoAGAAAAExLw8PHxc/Gh8PByERA0QREhMTFBQUFRYWFhcXFxgYFRUVFBUUExQT

KCgoKCgoJCQgIBwYGBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAgHBgYFBAMDAftkAV6W/K4
BAwUHCAoMDQ4PERETExQUCwsVFBN2dgkKCgoVFhQUExMREQ8ODQwKCAcFAwEBAwUHCAoMDQ4PERE
TExQUFBQTExERDw4NDAoIBwUDAQEEBgd2AV6W/ZYFBAMCAwEBAwUHCAoMDQ4PERETExQUFBQTExE
RDw4NDAoIBwUD1AoKCgkJCAgHBgYFBAMDAQEBAQMDBAUGBgcICAkJCgoKCgoKCQkICAcGBgUEAwM
BAQEBAwMEBQYGBwgICQkKCgEiBQUFAwMCAgICAwMFBQUFBQUDAwICAgIDAwUFAScKCgoJCQgIBwY
GBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAgHBgYFBAMDAQEBAQMDBAUGBgcICAkJCgqgZAF
eMpYKChQTExERDw4NDAoIBwUDAQEEBgd2dgUEAwIDAQEDBQcICgwNDg8RERMTFBQUFBMTEREPDg0
MCggHBQMBAQMFBwgKDA0ODxEREhMUFAsLFRQTdv6iMgJqCQoKChUWFBQTExERDw4NDAoIBwUDAQE
DBQcICgwNDg8RERMTFAADAAAAAANXA7UAIgBFAJMAAAEzHw4PDisBNRMzHw4PDisBNQMhPxEvDz8
PLxghAkgKCgkJCAgHBwYGBAQEAgEBAQEDAwQFBgYHBwgJCAkKCeDACgoJCQgIBwcGBgQEBAIBAQE
BAgQEBAYGBwcICAkJCgrAwAHDDQwMDBcWFRMSEQ8NDAoHBgQBAQIDBAYHBwkKCgsNDA4ODwsLCgo
KCAgIBgYFBQMDAQEBAQECAwQEBAUGDA8QEhQVFgwMDA0NDQ0N/nABogICAwQEBgYGBwgICQkKCQo
KCQgJBwgGBgUFBAMCArsBdwICAwQEBgYGCAcICQkKCQoKCQkIBwgGBgYEBAMCArv9MQEBAQIGCAo
MDg8REhQUFhcYGBERERAQEA4ODgwMDAoJCQcICQkKCgoLDAsMDAwMDQwNDQwNDQwMCwwLCxQUERA
ODQoFAwQDAgEBAQAABQAAAAAD9APzAAMABwALAA8AEwAANyE1ITUhNSE1ITUhNSE1ITUhNSEMA+j
8GAPo/BgD6PwYA+j8GAPo/BgMXn1enF59XX1eAAAAAAEAAAAAA9QD1ADUAAATHx8/DxcRIRcPDy8
fPx8fDzMvHw8eKwECAwQFBggICQoMDA0ODhAQERISExQUFRUWFhcXGBgYGBgXFxcWFhUVFBQTEhI
REIr+ZrsMDA0ODg4PEBAQEBESERISEhIREhEQEQ8QDw8ODg0NDAwLCgoJCQgHBgYEBAQCAQEBAQI
EBAQGBgcICQkKCgsMDA0NDg4PDxAPERAREhESEhwcGxoaGBgWFRQSEQ8OCwp7BQYHCAgJCQoLCww
NDQ4ODg8QEBEREBISEhMTFBMUFRQYGBgXFxYWFRUUFBMSEhQEA4ODQwMCgkICAYFBAMCAgAYGBc
XFxYWFRUUFBMSEhQEA4ODQ0LCgoICAYFBAMCAQECAwQFBggICgoLDQ0ODhCKAZq7DAsLCgkJCAc
HBQUEAwMBAQEBAgQEBAYGBwgICgkLCwwMDQ0ODg8PDxAREBESERISEhIREhEQERAPDw8ODg0NDAw
LCwkKCAgHBgYEBAQCAQECAwUICQsNDxASExUWFxgaExITERIREBAQDw8ODg0NDAsLCgoJCAcHBgY
EBAMCAQEBAgMEBQYICAoKCw0NDg4QEBESEhMUFBUVFhYXFxcYAAAAAgAAAAD8gP0AGcA7gAAARU
PGC8YPQE/FzsBHxcFHx8/DxcVATcBIyc/Dj0BLx0rAQ8dAoABAgIDAwQFBQUNDxATExYLCwwMDAw
NDQ0NDQ0NDQwNDAsMCxUUEhAPDQUFBQQDAwMBAQEBAwMDBAUFBQ0PEBIUFQsMCwwNDA0NDQ0NDQ0
NDAwMDAsLFhMTEA8NBQUFBAMDAgIB/Y0BAQMDBAUGBggICQkLCwsNDA4ODg8QEBARERISEhMTExE
REBEQEBAQDw8ODg4ODA0OAR1W/uMuDgoKCQkIBwYGBgQEAwMCAQICAwQFBgcHCAkKCgsMDA0NDg8
PDxARERERER

```
DAgICAgMEBAYGBgcHCAkICQkCeAkJCQgICAcGBgUFAwMDAQEDAwMFBQYGBwgICAkJsv2EAnxbAgI
DBAUFBgYHCAgICQkAAAASAN4AAQAAAAAAAAABAAAAAQAAAAAAQAQAAEAAQAAAAAAAgAHABEAAQA
AAAAAAwAQABgAAQAAAAAABAAQACgAAQAAAAAABQALADgAAQAAAAAABgAQAEMAAQAAAAAACgAsAFM
AAQAAAAAACwASAH8AAwABBAkAAAACAJEAAwABBAkAAQAgAJMAAwABBAkAAgAOALMAAwABBAkAAwA
gAMEAAwABBAkABAAgAOEAAwABBAkABQAWAQEAAwABBAkABgAgARcAAwABBAkACgBYATcAAwABBAk
ACwAkAY8gdG9vbGJhci1tYXRlcmlhbFJlZ3VsYXJ0b29sYmFyLW1hdGVyaWFsdG9vbGJhci1tYXR
lcmlhbFZlcnNpb24gMS4wdG9vbGJhci1tYXRlcmlhbEZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAdABvAG8AbABiAGEAcgAtAG0
AYQB0AGUAcgBpAGEAbABSAGUAZwB1AGwAYQByAHQAbwBvAGwAYgBhAHIALQBtAGEAdABlAHIAaQB
hAGwAdABvAG8AbABiAGEAcgAtAG0AYQB0AGUAcgBpAGEAbABWAGUAcgBzAGkAbwBuACAAMQAuADA
AdABvAG8AbABiAGEAcgAtAG0AYQB0AGUAcgBpAGEAbABGAG8AbgB0ACAAZwBlAG4AZQByAGEAdAB
lAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHU
AZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAAAAAAKAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAjAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4BDwEQAREBEgE
TARQBFQEWARcBGAEZARoBGwEcAR0BHgEfASABIQEiASMBJAAQVGV4dF9PdXRkZW50XzAwMQtQaWN
0dXJlXzAwMQxTZXR0aW5nc18wMDEQQ29sb3JfcGlja2VyXzAwMhBBbGlnbl9DZW50ZXJfMDA2CEx
pbmVfMDAxDVVuZGVybGluZV8wMDEMU29ydF9aLUFfMDAxCFVuZG9fMDAxEENoYXJ0X2J1YmJsZV8
wMDELRG93bmxvYWRfMDAPVGV4dF9pbmRlbnRfMDAxEkNoYXJ0X0RvdWdobnV0XzAwMQlDbGVhcl8
wMDINTnVtYmVyaW5nXzAwMQxTb3J0X0EtWl8wMDEKSXRhbGljXzAwMQtCdWxsZXRzXzAwMQlQYXN
0ZV8wMDEIUmVkb18wMDEPQ2hhcnRfcmFkYXJfMDAxD0FsaWduX1JpZ2h0XzAwMQlUYWJsZV8wMDE
OQWxpZ25fTGVmdF8wMDEITWVudV8wMDEHQ3V0XzAwMghCb2xkXzAwMRFBbGlnbl9KdXN0aWZ5XzA
wMQpSZWxvYWRfMDAxClNlYXJjaF8wMDEKVXBsb2FkXzAwMQpEZXNpZ25fMDA1CkV4cG9ydF8wMDE
IQ29weV8wMDIAAA==) format("truetype");
font-weight: normal;
font-style: normal;
}
.e-cut-icon:before {
content: "\e719";
}
.e-copy-icon:before {
content: "\e721";
}
.e-paste-icon:before {
content: "\e712";
}
.e-color-icon:before {
content: "\e703";
}
.e-bold-icon:before {
content: "\e71a";
}
.e-underline-icon:before {
content: "\e706";
}
.e-italic-icon:before {
content: "\e710";
}
.e-ascending-icon:before {
content: "\e70f";
}
.e-descending-icon:before {
content: "\e707";
}
.e-clear-icon:before {
content: "\e70d";
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Run the project

To run the project, use the following command:

`bash

npm run dev

`

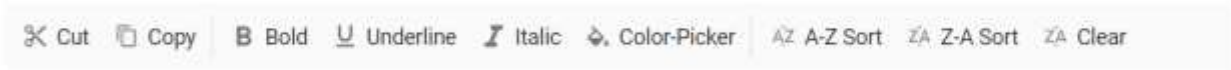
or

`bash

yarn run dev

`

The output will appear as follows:



Refer the sample Vue 3 using Composition API Toolbar getting started

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Item configuration in Vue Toolbar component

The Vue Toolbar can be rendered by defining an array of items. Items can be constructed with the following built-in command types or item template.

Button

`Button` is the default command `type`, and it can be rendered by using the `text` property. Properties of the button command type:

Property | Description

text | The text to be displayed for button.

id | The ID of the button to be rendered. If the ID is not given, auto ID is generated.

prefixIcon | Defines the class used to specify an icon for the button. The icon is `positioned before` the text if text is available or the icon alone button is rendered.

suffixIcon | Defines the class used to specify an icon for the button. The icon is `positioned after` the text if text is available. If both `prefixIcon` and `suffixIcon` are specified, only `prefixIcon` is considered.

width | Used to set the width of the button.

Separator

The `Separator` type adds a vertical separation between the Toolbar's single/multiple commands.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:100%;">
        <br>
         <ejs-toolbar >
    <e-items>
             <e-item text='Cut'></e-item>
             <e-item text='Copy'></e-item>
             <e-item text='Paste'></e-item>
             <e-item type='Separator'></e-item>
             <e-item text='Bold'></e-item>
             <e-item text='Italic'></e-item>
             <e-item text='Underline'></e-item>
          </e-items>
    </ejs-toolbar>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(ToolbarPlugin);
export default {
  name: 'app',
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/toolbar/toolbar-items-cs1" %}

If Separator is added as the first or the last item, it will not be visible.

Enabling tab key navigation in Toolbar

The tabIndex property of a Toolbar item is used to enable tab key navigation for the item. By default, the user can switch between items using the arrow keys, but the tabIndex property allows you to switch between items using the Tab and Shift+Tab keys as well.

To use the tabIndex property, you need to set it for each Toolbar item that you want to enable tab key navigation. The tabIndex property should be set to a positive integer value. A value of 0 or a negative value will disable tab key navigation for the item.

For example, to enable tab key navigation for two Toolbar items, you can use the following code:

`

<template>

<ejs-toolbar>

```
<e-items>
<e-item text='Item 1' tabIndex="1"></e-item>
<e-item text='Item 2' tabIndex="2"></e-item>
<e-item text='Item 3' tabIndex="3"></e-item>
</e-items>
</ejs-toolbar>
</div>
</template>
<script>
import Vue from 'vue';
import { ToolbarPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(ToolbarPlugin);
export default {
name: 'app',
}
</script>
```

With the above code, the user can switch between the two Toolbar items using the Tab and Shift+Tab keys, in addition to using the arrow keys. The items will be navigated in the order specified by the tabIndex values.

If you set the tabIndex value to 0 for all Toolbar items, tab key navigation will be based on the element order rather than the tabIndex values. For example:

```
<template>
<div id="app">
<ejs-toolbar>
<e-items>
<e-item text='Item 1' tabIndex="0"></e-item>
<e-item text='Item 2' tabIndex="0"></e-item>
<e-item text='Item 3' tabIndex="0"></e-item>
</e-items>
</ejs-toolbar>
</div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

<script>

import Vue from 'vue';

import { ToolbarPlugin } from '@syncfusion/ej2-vue-navigations';

Vue.use(ToolbarPlugin);

export default {

name: 'app',

}

</script>

`

In this case, the user can switch between the two Toolbar items using the Tab and Shift+Tab keys, and the items will be navigated in the order in which they appear in the DOM.

Example:

Here is an example of how you can use the tabIndex property to enable tab key navigation for a Toolbar component:

APP.VUE

```
<template>
  <div id="app">
    <ejs-toolbar>
      <e-items>
        <e-item text='Cut' tabIndex="0"></e-item>
        <e-item text='Copy' tabIndex="0"></e-item>
        <e-item text='Paste' tabIndex="0"></e-item>
        <e-item type='Separator'></e-item>
        <e-item text='Bold'></e-item>
        <e-item text='Italic'></e-item>
        <e-item text='Underline'></e-item>
      </e-items>
    </ejs-toolbar>
  </div>
</template>
<script>
import Vue from 'vue';
import { ToolbarPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(ToolbarPlugin);
export default {
  name: 'app',
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/toolbar/toolbar-container-cs2" %}

With the above code, the user can switch between the Toolbar items using the Tab and Shift+Tab keys, and the items will be navigated based on the element order.

See Also

- How to set item wise custom template

Template configuration in Vue Toolbar component

Integrate menu component

You can integrate menu component as toolbar item in Toolbar using Vue **template**. Menu can be populated with items as needed.

APP.VUE

```
<template>
    <div id="app">
        <ejs-toolbar >
           <e-items>
             <e-item :template='menuTemplate' ></e-item>
          </e-items>
        </ejs-toolbar>
    </div>
</template>
<script>
import Vue from 'vue';
import { MenuPlugin, ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
Vue.use(MenuPlugin);
Vue.use(ToolbarPlugin);
var menutemplateVue = Vue.component("demo", {
  template: '<ejs-menu :items="menuItems"></ejs-menu>',
  data() {
    return {
      data: {},
      menuItems: [
            {
                text: 'Appliances',
                items: [
                    {
                        text: 'Kitchen',
                        items: [
                            { text: 'Electric Cookers' },
                            { text: 'Coffee Makers' },
                            { text: 'Blenders' }
                        ]
                    },
                    {
                        text: 'Washing Machine',
                        items: [
                            { text: 'Fully Automatic' },
                            { text: 'Semi Automatic' }
                        ]
                    },
```

```
                    {
                        text: 'Air Conditioners',
                        items: [
                            { text: 'Inverter ACs' },
                            { text: 'Split ACs' },
                            { text: 'Window ACs' }
                        ]
                    }
                ]
            },
            {
                text: 'Accessories',
                items: [
                    {
                        text: 'Mobile',
                        items: [
                            { text: 'Headphones' },
                            { text: 'Memory Cards' },
                            { text: 'Power Banks' }
                        ]
                    },
                    {
                        text: 'Computer',
                        items: [
                            { text: 'Pendrives' },
                            { text: 'External Hard Disks' },
                            { text: 'Monitors' }
                        ]
                    }
                ]
            },
            {
                text: 'Fashion',
                items: [
                    {
                        text: 'Men',
                        items: [
                            { text: 'Shirts' },
                            { text: 'Jackets' },
                            { text: 'Track Suits' }
                        ]
                    },
                    {
                        text: 'Women',
                        items: [
                            { text: 'Kurtas' },
                            { text: 'Salwars' },
                            { text: 'Sarees' }
                        ]
                    }
                ]
            },
            {
                text: 'Home & Living',
                items: [
                    {
                        text: 'Furniture',
```

```
                        items: [
                            { text: 'Beds' },
                            { text: 'Mattresses' },
                            { text: 'Dining Tables' }
                        ]
                    },
                    {
                        text: 'Decor',
                        items: [
                            { text: 'Clocks' },
                            { text: 'Wall Decals' },
                            { text: 'Paintings' }
                        ]
                    }
                ]
            }
        ],
    };
  }
});
export default {
   data: function() {
    return {
        menuTemplate: function() {
            return {
                template: menutemplateVue
            }
        }
    }
  },
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/toolbar/menu-component-cs1" %}

Responsive mode in Vue Toolbar component

This section explains the supported display modes of the Toolbar when the content exceeds the viewing area. Possible modes are:

- Scrollable
- Popup

Scrollable

The default overflow mode of the Toolbar is Scrollable. Scrollable display mode supports display of commands in a single line with

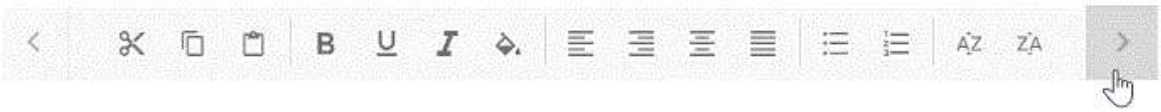
horizontal scrolling enabled when commands overflow to available space.

Copyright © 2001 -2024 Syncfusion Inc.

- The right and left navigation arrows are added to the start and end of the Toolbar to navigate to hidden commands.
- You can also see the hidden commands using touch swipe action.
- By default, if navigation icon in the left side is disabled, you can see the hidden commands by moving to the right.
- By clicking the arrow or by holding the arrow continuously, hidden commands will become visible.
- If device navigation icons are not available, you can touch swipe to see the hidden commands of the Toolbar.

- Once the Toolbar reaches the last or first command, the corresponding navigation icon will be disabled and you can move to the opposite direction.

- You can continuously scroll the Toolbar content by holding the navigation icon.



APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:100%;">
        <br>
        <ejs-toolbar >
            <e-items>
             <e-item text='Cut' prefixIcon = 'e-cut-icon tb-icons'></e-item>
             <e-item text='Copy' prefixIcon = 'e-copy-icon tb-icons'></e-
item>
             <e-item text='Paste' prefixIcon = 'e-paste-icon tb-icons'></e-
item>
             <e-item type='Separator'></e-item>
             <e-item text='Bold' prefixIcon = 'e-bold-icon tb-icons'></e-
item>
             <e-item text='Underline' prefixIcon = 'e-underline-icon tb-
icons'></e-item>
             <e-item text='Italic' prefixIcon = 'e-italic-icon tb-
icons'></e-item>
             <e-item type='Separator'></e-item>
             <e-item text='A-Z Sort' prefixIcon = 'e-ascending-icon tb-
icons'></e-item>
             <e-item text='Z-A Sort' prefixIcon = 'e-descending-icon tb-
icons'></e-item>
             <e-item text='Clear' prefixIcon = 'e-clear-icon tb-icons'></e-
item>
            </e-items>
        </ejs-toolbar>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(ToolbarPlugin);
export default {
  name: 'app',
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.custom_bold .e-tbar-btn-text {
    font-weight: 900;
}
.custom_italic .e-tbar-btn-text {
    font-style: italic;
}
.custom_underline .e-tbar-btn-text {
    text-decoration: underline red;
}
.e-txt-casing .e-tbar-btn-text {
    font-variant: small-caps;
}
.e-tbar-btn .e-icons {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    font-family: 'Material_toolbar';
    font-size: 16px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.custom_bold .e-tbar-btn-text {
    font-weight: 900;
}
.custom_italic .e-tbar-btn-text {
    font-style: italic;
}
.custom_underline .e-tbar-btn-text {
    text-decoration: underline red;
}
.e-txt-casing .e-tbar-btn-text {
    font-variant: small-caps;
}
@font-face {
    font-family: "Material_toolbar";
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tShMAAAEoAAAAVmNtYXDoMOjqAAACDAAAAHhnbHlmIuy
19QAAAswAACNMaGVhZA6okZMAAADQAAAANmhoZWEIUQQkAAAArAAAACRobXR4jAAAAAAAAYAAAAC
MbG9jYYc0kUIAAAKEAAAASG1heHABOwG8AAABCAAAACBuYW1lx/RZbQAAJhgAAAKRcG9zdJZeEVU
AACisAAACGAABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAIwABAAAAAQAAAQsu/F8
PPPUACwQAAAAAANXLJlEAAAAA1csmUQAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAjAbAADgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnIQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQ
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAA
AAAQAAAAEAAAABAAAAAQAAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAGQAAAAEAAQAAQAA5yH//wA
A5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcACAAJAAoACwAMAA0ADgAPABAAEQASABMAFAAVABY
AFwAYABkAGgAbABwAHQAeAB8AIAAhACIAAAAAADIAjgFwAfgCIAKYAxIDSAO2BRYFMAVcBnIGugb
2ByoHQgguCNYJRgn6CiQKiAquCsgMFgzADOYNzg7WDvAQyBEyEaYABwAAAAAD9APzAAMABwAKAA4
AEgAVABkAADchNSElITUhJTkBBSE1ITUhNSEFFxEnITUhDAPo/BgBtgIy/c7+SgG2AjL9zgIy/c7
+Svr6A+j8GAxefV67Pl19Xvr6AfScXgAAAAIAAAAAA/QD9AAEAEgAACUhNxc3AREfDyE/DxEvDyE
PDgOF/PbDisP9gQEBAwQEBgYICAgJCgoLCwsDCgsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQo
KCwsL/PYLCwsKCgkICAgGBgQEAwGz+qf6AYX89gsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQo
KCwsLAwoLCwsKCgkICAgGBgQEAwEBAQEDBAQGBggICAkKCgsLAAACAAAAAAPzA/QAQAC/AAABFQ8
PLw8/Dx8OAQ8ELwErAQ8FFR8FBxcPAxUfBzsBNx8LOwI/Cx8BOwE/Bj0BLwQ/Aic/BC8HKwEHLws
rAg8FArIBAgUGBwkKDAwODxAQERITEhIREQ8PDgwMCgkHBgUCAQECBQYHCQoMDA4PDxEREhITEhE
QEA8ODAwKCQcGBQL+zxUWFhUWfwUFBAUDBANqAgEBAgIDbgMDbwMCAQEBAmkDBAQEBQSEFBYWFxQ
CAgIDBAQEBcwFBAQEAwICAhQXFRYVgAQFBQQEAwNoAgEBAgIDcAEBAQNvAgIBAQEBA2gDBAQFBAW
DFBYWFxIBAgMDAwQFBcwFBAQDBAICAgAJCRIQEBAODgwLCgkHBgQDAQEDBAYHCQoLDA4OEBAQEhI
SEhAQEA4ODAsKCQcGBAMBAQMEBgcJCgsMDg4QEBASAc6ECwwNDjIBAQICA7QEBQQFBAMEUjIyVgM
EBAQFBAWwAwICATMODQwLhAQEBAMCAgECAgIDBAMEhAsMDQ4yAQECAgOwBAQFBAUEAwRSDAwaMlY
DBAQEBQQFsAMCAgEzDg0MC4QEAwQDAgICAgICAwQDAAAAAAMAAAAAA/MD2AAyADUAaQAAJRUfDTs
BPw41LwgPBwMhAScXAQ8GHQEfBQEfBjsBPwYBPwYvBwEDFgIDBAQGBgcICQkKCgsLCwsLCwoKCQg
```

ICAYGBAQDAQEDBAcMCQoLFCMtFQoJCQcFBHv96gEL04X+4gYFBAQCAgICAgIEBAUBNwcHBwgHCAc
IBwgHCAgHBwYBOAUEAwMCAQEBAQIDAwQFBv4PlwsLCwoKCQkIBwYGBAQDAgIDBAQGBgcICQkKCgs
LCwcPEBAYEBAPHCk3HRAQEBAQEAEIAQrThf7iBgcIBwgHCAgICAgHBwcH/skGBQQEAwIBAQIDBAQ
FBgE3BwcHBwgICAgHCAgHCAcGAfEABQAAAAAD9APzAAMABwALAA8AEwAANyE1ITchNSEnITUhNyE
1ISchNSEMA+j8GN4CLP3U3gPo/BjeAiz91N4D6PwYDF6AW5xefVqAXgAAAAAEAAAAAAP0A/QACQA
TABcAWwAAAQcVMzcXMzUjLwEjFTMbATM1IwElESERBxEfDyE/DxEvDyEPDgFro8ObnnROxOp0nZv
qTij+8AGW/NReAQEDBAQGBggICAkKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJCgo
LCwv89gsLCwoKCQgICAYGBAQDAQENAyOWliO4BSUBK/7VJQFSffzUAywR/PYLCwsKCgkICAgGBgQ
EAwEBAQEDBAQGBggICAkKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJCgoLCwAAAAA
CAAAAAAOWA/QAAwBpAAA3ITUhExUfHTsBPx01ESMRDw8vDxEjagMs/NRKAgIDAwUFBgcHCAkJCgs
LCwwNDQ0ODw4PEA8QERARERERERBEQDxAPDg8ODQ0NDAsLCwoJCQgHBwYFBQMDAgKLAQMFBggKCww
ODxARERMTFBQTExEREA8ODAsFCQcGBAKLDH0BsBERERERQEA8QDg8ODg0MDQsLCwoJCQgHBwYFBQQ
DAgEBAgMEBQUGBggICQkKCgsMDAwNDg4ODw8PEBAQERERAb7+RRQTEhIREA8NDQsKCAYFAwEBAwU
GCAoLDQ0PCBASEhMTAcUABQAAAAAD9APXAAIABQANABcAGgAAJTcjASM3ATM3MxczAyMFIQEVITU
hATUhJTMnAgJx4wG/vl/+/Fot+i1a3FD9RgEg/t4Bov7UAST+aAF/6XQobQET//47eHgCM07+XD5
NAaQ/UHMAAAAAAQAAAAAD9ALoAF8AABMhJz8PHxo3Lx8PDycMAbWyDQ0ODg8PDxAQEBERERIREhA
QEBAQDw8PDw4ODg0NDQwMFxYTEhAHBgYGBXUHBwgJCQoLCwwNDQ0PDg8QEBERERITEhMUExQVFBU
VFRgYFxcXFhYVFhQUFBMTEhGwARi6CwsJCggICAYGBgQEAwIBAQEBAgIDBAQFBQYHBwgICQkKFRY
YGhsODg8PDygUFBMTEhISERARDw8PDg0NDAsLCgoICAgGBgQEAwMBAQECAwQFBgcJCQoLDA0ODg+
6AAYAAAAAA/MD9AA/AGsAqwDrAO8BMwAAARUfDTsBPw09AS8ODw4lHwk7AT8IPQEvByMnByMPByU
fDz8PLw8PDiUfDz8OPQEvDSsBDw0lESERBxEfDyE/DxEvDyEPDgHhAgMFBQYHCAkKCgsLDA0NDA0
MCwsKCgkIBwYFBQMCAgMFBQYHCAkKCgsLDA0MDQ0MCwsKCgkIBwYFBQMC/scBAQEFBwgKCwYGBwY
GBgwKCAcFAQEBAQUHCAoMBgYGBwYGCwoIBwUBAQHzAQECBAQEBgYGCAcICQkJCgoJCQgJBwgGBgY
EBAMDAQEBAQMDBAQGBgYIBwkICQkKCgkJCQgHCAYGBgQEBAIB/qgBAQMEBAYGBwgICQoKCgsLCws
LCgkJCQcHBwUFAwMCAgMDBQUHBwcJCQkKCwsLCwsKCgoJCAgHBgYEBAMBAlD81F4BAQMDBAUGBgc
HCAkJCQkKAyYKCQkJCQgHBwYGBQQEAgEBAQECBAQFBgYHBwgJCQkJCvzaCgkJCQkIBwcGBgUEAwM
BAWQNDAwMCwoKCQgHBgUFAwICAwUFBgcICQoKCwwMDA0NDAwLCwsJCQgHBwUEAwIBAQIDBAUHBwg
JCQsLCwwMMQYGBgsKCQcEAgEBAgQHCQoLBgYGBwYGCwoJBgUCAQECBQYJCgsGBvMJCgkICAgHBwY
FBQQDAwEBAQEDAwQFBQYHBwgICAkKCQoJCQkICAcGBwUFBAMCAQEBAQIDBAUFBwYHCAgJCQkGCws
KCgoJCAgHBgYEBAMBAQEBAwQEBgYHCAgJCgoKCwsLCwsKCQkJBwcHBQUDAwICAwMFBQcHBwkJCQo
LC9/81AMsA/zaCgkJCQkIBwcGBgUEBAIBAQEBAgQEBQUHBwcICQkJCQoDJgoJCQkJCAcHBwUFBAQ
CAQEBAQIEBAUFBwcHCAkJCQkAAAAAAgAAAAADtQP0AAMACgAANyE1IQEjCQEjESFKA2z8lAEG7AG
cAZzs/qAMfQIK/mQBnAFhAAYAAAAAA/QD8wADAAcACwAPABIAFgAANyE1ISUhNSE1ITUhNSE1KQE
RNwMhNSEMA+j8GAG2AjL9zgIy/c4CMv3O/kr6+gPo/BgMXn1efV19Xv4M+gGWXgAFAAAAAAPzA/M
AJQBpAKgArADwAAABFT8bIw8GBR8PNSMvDT8CJw8OHw8TByMPDBc/AzsBHwUzHwYzLxcjJREhEQc
RHw8hPw8RLw8hDw4CKg8QDw8ODg4NDQwMDAwKCwoKCQgJDw0KCQQDAgLxBAUGBggJC/7WDQ0NDg4
PDw8PEA8QEBAQEAoKCQkJCQgIBgQEBAUDAQEDBKsKCQgIBwYGBQUDAwMBAQEBAgQEBQYGBwICQkJCQo
DJgoJCQkJCAcHBgUFBAQCAQEBAQIEBAUFBwcHCAkJCQkAAgAAAAAD8wPrAB8AMwAAEw8HHww/BBU
hNSEBNwkBPwcvCDYKCAcGBw8PEA8OBwMEBQYIBQcGBw8PEA8QEBAQEAoKCQkJCQgIBgQEBAUDAgE
BAgMFBgcJCtMBoQsMDQ0NDg4ODg0ODQ0NDAuvBgUDAQECBAY0I14BJEH+xAGRCwwMDQ0ODg4ODg4
NDQwMC9QABgAAAAAD9AP0AAMADwATAB0AIQAnAAAlITUhIzMVIxUzFSMVMzUjNyE1ISMzBxUzNSM
3NSM3ITUhJzMVMzUjAQYC7v0S+n0/P328vPoC7v0S+nh4vHh4vPoC7v0S+j4/fWpeID4gPvrbXXw
/P3w/u14gvPoAAAABQAAAAAD9APbAAIABQANABcAGgAAJTcjAyM3ATM3MxczAyMFIQEVITUhATU
hJzMnAgJx4x6+Xv76Wi39LF3fTwFlAST+3AGk/tIBJP5mw+l0OJXMBGP/+NhJXMBGP/+NhJXMBxU
fDj8OPQEvDg8OAQYC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgYGBQUEBAMCAQECAwMEBAUFBgY
HBwgICAgICQkKCgsLCwwMDQ0NDAwLCwoKCQgICAYGBgQEAwMBAvoC7v0S+gIBAwMEBQUFBgcGCAc
ICAgICAcHBgYGBQUEBAMCAQECAwMEBAUFBgYHBwgICAgICQkKCgsLCwwMDQ0NDAwLCwoKCQgICAY
GBgQEAwMBAvoC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgYGBQUEBAMCAQECAwMEBAUFBgYHBwg
ICAgICQkKCgsLCwwMDQ0NDAwLCwoKCQgICAYGBgQEAwMBAkpeLwgIBwcHBwYFBQUDBAICAQE
BAQICBAMFBQUGBwcHBwgICAgICQkKCgsLCwwMDQ0NDAwLCwoKCQgICAYGBgQEAwMBAgYFBAQDAwIBAQEBAgMDBAQFBgYGBwcICA

GBgUEBAMDAgEBAgMDBAQFBgYGBwcICAgICAcHBwcGBQUFAwQCAgEBAQECAgQDBQUFBgcHBwcIAUB
dLggICAcHBgYGBQQEAwMCAQEBAQIDAwQEBQYGBgcHCAgICAgHBwcHBgUFBQMEAgIBAQEBAgIEAwU
FBQYHBwcHCAAAAwAAAAADmQP0AAcAKACNAAABFSE1MxEhESUHFQ8GLwc/Bx8GJysBDw0VERUfDTM
hMz8NNRE1Lw0rAS8OKwEPDQEdAcZb/YQBbAEDBAYHBwkJCQkHBwYEAwEBAwQGBwcJCQkJBwcGBAO
svwkJCQgICAcGBgYEBAMCAgICAwQEBgYGBwgICAkJCQJ8CQkJCAgIBwYGBgQEAwICAgIDBAQGBgY
HCAgICQkJvwMFBQYGBwgICQkJCgoKCwsLCwoKCgkJCQgIBwYGBQUDPoiI/SkC1y4FBQgIBwUEAwE
BAwQFBwgICgkICAcFBQIBAQIFBQcICCQCAgMEBAYGBgcICAgJCQn9KQkJCQgICAcGBgUFBAMCAgI
CAwQFBQYGBwgICAkJCQLXCQkJCAgIBwYGBgQEAwICCgkJCAgIBwYGBQQEAwICAgIDBAQFBgYHCAg
ICQkAAQAAAAAD9ALoAGAAAAExLw8PHxc/Gh8PByERA0QREhMTFBQUFRYWFhcXFxgYFRUVFBUUExQ
TEhMSEREREBAPDg8NDQ0MCwsKCQkIBwd1BQYGBgcQEhMWFwwMDQ0NDg4ODw8PDxAQEBAQEhESERE
QERAQDw8PDg4NDbIBtQIuDw4ODQwLCgkJBwYFBAMCAQEBAwMEBAYGCAgICgoLCwwNDQ4PDw8REBE
SEhITExQUKA8PDw4OGxoYFhUKCQkICAcHBgUFBAQDAgIBAQEBAgMEBAYGBggICAoJCwu6AdAAAAA
OAAAAAAP0A/MAAgAFAAgACwAQABQAFwAbAB4AIQApAC0AMQB1AAABETclFzUXNyMFNyETFQUhEQE
hJRMlMycFMSEnBzcnBxcRBRMDBSUDEy0BEQMlIwUDEQcRHw8hPw8RLw8hDw4CGcj+ZaG3MJb+wM7
+4jQBCv6EAy7+ggEKdP1S3JkBCwEjWemWlvrIATJ0dP7n/up3dwEWAZhy/vQ0/vZyXgEBAwQEBgY
ICAgJCgoLCwsDCgsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQoKCwsL/PYLCwsKCgkICAgGBgQ
EAwEBxv7fWSQ730Bky8v+9QNxAYH+f28BHx2ZmcukmTgJywEeP/7n/ud3dwEZARl3Bv5xAR1ycv7
yAYAR/PYLCwsKCgkICAcHBQUEAwEBAQEDBAUFBwcICAkKCgsLCwMKCwsLCgoJCAgHBwUFBAMBAQE
BAwQEBgYIBwkJCgoLCwAAAAAFAAAAAAP0A/MAAwAHAAsADwATAAA3ITUhJSE1ISUhNSElITUhJSE
1IQwD6PwYAVgCkP1w/qgD6PwYAVgCkP1w/qgD6PwYDF6AW5xefV19XgAAAAAKAAAAAAP0A/MAAwA
HAAsADwATABcAGwAfACMARwAAARUjNSMVIzUjFSM1ARUjNSMVIzUjFSM1JRUjNSMVIzUjFSM1JxE
fByE/BxEvByEPBgOW+j7bP9oDLPo+2z/aAyz6Pts/2l4BAwUGAwgJCgOJCgkJBwYDBAIBAwUGAwg
JCvx3CgkJBwYFAwElvb27u7u7ARrb29vb29v6vLy8vLy8hvyCCwoJBwQGBAIBAwUHBwUJCgOECwo
JBwQGBAIBAwUGCAkKAAAAAAUAAAAAA/QD8wADAcACwAPABMAADchNSE1ITUhNSE1ITUhNSE1ITU
hDAPo/BgCkP1wA+j8GAKQ/XAD6PwYDF6BV59efVqAXgAAAADAAAAAAP0A00AAwAHAAsAADchNSE
1ITUhNSE1IQwD6PwYA+j8GAPo/Bizb6Zwpm8AAAAABQAAAAD9AP0AD8AxwCfAKQBIgAAJQ8PLw8
/Dx8OExUPBSsBLwU9AT8FOwEfBQMPDy8PPw8fEAE1IwUVHw8zPwMXBy8FDw8fDz8PNS8DNwEzNQE
/BS8PDw4BOAEBAwMEBQYGBwgICQkKCgoKCgoJCQgIBgYFBAMDAeICAgMDBQUFBQUFAwMCAgICAwMFBQUFBQUDAwIC4QEBAwMEBQYGBwgICQk
KCgoKCgoJCQgIBgYFBAMDAftkAV6W/K4
BAwUHCAoMDQ4PERETExQUCwsVFBN2dgkKCgoVFhQUExMREQ8ODQwKCAcFAwEBAwUHCAoMDQ4PERE
TExQUFBQTExERDw4NDAoIBwUDAQEEBgd2AV6W/ZYFBAMCAwEBAwUHCAoMDQ4PERETExQUFBQTExE
RDw4NDAoIBwUD1AoKCgkJCAgHBgYFBAMDAQEBAQMDBAUGBgcICAkJCgoKCgoKCQkICAcGBgUEAwM
BAQEBAwMEBQYGBwgICQkKCgEiBQUFAwMCAgICAwMFBQUFBQUDAwICAgIDAwUFAScKCgoJCQgIBwY
GBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAgHBgYFBAMDAQEBAQMDBAUGBgcICAkJCgqgZAF
eMpYKChQTExERDw4NDAoIBwUDAQEEBgd2dgUEAwIDAQEDBQcICgwNDg8RERMTFBQUFBMTEREPDg0
MCggHBQMBAQMFBwgKDA0ODxEREXMUFAsLFRQTdv6iMgJqCQoKChUWFBQTExERDw4NDAoIBwUDAQE
DBQcICgwNDg8RERMTFAADAAAAAAANXA7UAIgBFAJMAAAEzHw4PDisBNRMzHw4PDisBNQMhPxEvDz8
PLxghAkgKCgkJCAgHBwYGBAQEAgEBAQEDAwQFBgYHBwgJCAkKCeDACgoJCQgIBwcGBgQEBAIBAQE
BAgQEBAYGBwcICAkJCgrAwAHDDQwMDBcWFRMSEQ8NDAoHBgQBAQIDBAYHBwkKCgsNDA4ODwsLCgo
KCAgIBgYFBQMDAQEBAQECAwQEBAUGDA8QEhQVFgwMDA0ODQ0N/nABogICAwQEBgYGBwgICQkKCQo
KCQgJBwgGBgUFBAMCArsBdwICAwQEBgYGCAcICQkKCQoKCQkIBwgGBgUEBAMCArv9MQEBAQIGCAo
MDg8REhQUFhcYGBERERAQEA4ODgwMDAoJCQcICQkKCgoLDAsMDAwNDQwNDQwNDQwMCwwLCxQUERA
ODQoFAwQDAgEBAQAABQAAAAAD9APzAAMABwALAA8AEwAANyE1ITUhNSE1ITUhNSE1ITUhNSEMA+j
8GAPo/BgD6PwYMXn1enF59XX1eAAAAAAEAAAAAA9QD1ADUAAATHx8/DxcRIRcPDy8
fPx8fDzMvHw8eKwECAwQFBggICQoMDA0ODg0NDAwLCgoJCQgIBwcGBgUFBAQDAgIBAQIDBggICQo
MDA0ODhAQERISERERISEhwGxoaGBgWFRQSEQ8OCwp7BQYHCAgJCQoLDAsMDAwNDQwNDQwNDQwMCwwLCxQUERA
ODQoFAwQDAgEBAQAABQAAAAD8APzAAMABwALAA8AEwAANyE1ITUhNSE1ITUhNSEMA+j
8GAPo/BgMXn1enF59XX1eAAAAAgAAAAAD8gP0AGcA7gAAARU
PGC8YPQE/FzsBHxcFHx8/DxcVATcBIyc/Dj0BLx0rAQ8dAoABAgIDAwQFBQUNDxATExYLCwwMDAw
NDQ0NDQ0NDQwNDAsMCxUUEhAPDQUFBQQDAwMBAQEBAwMDAUFBQ0PEBIUFQsMCwwNDA0NDQ0NDQ0
NDAwMDAsLFhMTEA8NBQUFBAMDAgIB/Y0BAQMDBAUGBggICQkLCwsNDA4ODg8QEBARERISEhMTExE
REBEQEBAQDw8ODg4ODA0OAR1W/uMuDgoKCQkIBwYGBgQEAwMCAQICAwQFBgcHCAkJCgoLCwsMDAw
NDg8

```
PDxARERERER
```

```
    content: "\e719";
  }

  .e-copy-icon:before {
    content: "\e721";
  }

  .e-paste-icon:before {
    content: "\e712";
  }

  .e-color-icon:before {
    content: "\e703";
  }

  .e-bold-icon:before {
    content: "\e71a";
  }

  .e-underline-icon:before {
    content: "\e706";
  }
  .e-italic-icon:before {
    content: "\e710";
  }
  .e-ascending-icon:before {
    content: "\e70f";
  }

  .e-descending-icon:before {
    content: "\e707";
  }

  .e-clear-icon:before {
    content: "\e70d";
  }
</style>
```

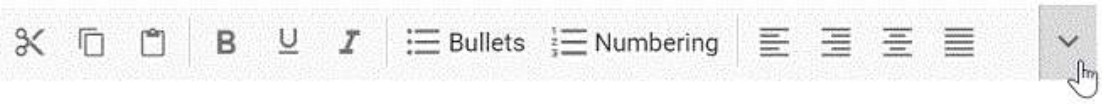
{% previewsample "page.domainurl/code-snippet/toolbar/responsive-mode/toolbar-scrollable-cs1" %}

Popup

Popup is another type of overflowMode in which the Toolbar container holds the commands that can be placed in the available space. The rest of the overflowing commands that do not fit within the viewing area moves to the overflow popup container.

The commands placed in the popup can be viewed by opening the popup using the drop down icon given at the end of the Toolbar.

Copyright © 2001 -2024 Syncfusion Inc.



If the popup content overflows the height of the page, then the rest of the commands will be hidden.

Priority of commands

Default popup priority is set as none, and when the commands of the Toolbar overflow, the ones listed last will be moved to the popup.

You can customize the priority of commands to be displayed on the Toolbar and popup by using the overflow property.

Property | Description

Both | Button text is visible in both Toolbar and Popup.

Overflow | Button text is only visible in Popup.

Toolbar | Button text is only visible on the Toolbar.

In the following code sample, text is only visible in the popup container and not in the Toolbar container.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:100%;">
        <br>
        <ejs-toolbar width='380' overflowMode='Popup'>
            <e-items>
              <e-item text='Cut' prefixIcon = 'e-cut-icon tb-icons' overflow
='Show' showTextOn = 'overflow'></e-item>
              <e-item text='Copy' prefixIcon = 'e-copy-icon tb-icons'
overflow ='Show' showTextOn = 'overflow'></e-item>
              <e-item text='Paste' prefixIcon = 'e-paste-icon tb-icons'
overflow ='Show' showTextOn = 'overflow'></e-item>
              <e-item type='Separator'></e-item>
              <e-item text='Bold' prefixIcon = 'e-bold-icon tb-icons'
overflow ='Show' showTextOn = 'overflow' ></e-item>
              <e-item text='Underline' prefixIcon = 'e-underline-icon tb-
icons' overflow ='Show' showTextOn = 'overflow' ></e-item>
              <e-item text='Italic' prefixIcon = 'e-italic-icon tb-icons'
overflow ='Show' showTextOn = 'overflow' ></e-item>
              <e-item text='Color-Picker' prefixIcon = 'e-color-icon tb-
icons' overflow ='Hide' showTextOn = 'overflow' ></e-item>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            <e-item type='Separator'></e-item>
            <e-item text='A-Z Sort' prefixIcon = 'e-ascending-icon tb-
icons' overflow ='Show' showTextOn = 'overflow'></e-item>
            <e-item text='Z-A Sort' prefixIcon = 'e-descending-icon tb-
icons' overflow ='Show' showTextOn = 'overflow'></e-item>
            <e-item text='Clear' prefixIcon = 'e-descending-icon tb-icons'
overflow ='Show' showTextOn = 'overflow'></e-item>
          </e-items>
        </ejs-toolbar>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(ToolbarPlugin);
export default {
  name: 'app',
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.custom_bold .e-tbar-btn-text {
    font-weight: 900;
}
.custom_italic .e-tbar-btn-text {
    font-style: italic;
}
.custom_underline .e-tbar-btn-text {
    text-decoration: underline red;
}
.e-txt-casing .e-tbar-btn-text {
    font-variant: small-caps;
}
.e-tbar-btn .e-icons {
    font-family: 'Material_toolbar';
    font-size: 16px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.custom_bold .e-tbar-btn-text {
    font-weight: 900;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.custom_italic .e-tbar-btn-text {
    font-style: italic;
}
.custom_underline .e-tbar-btn-text {
    text-decoration: underline red;
}
.e-txt-casing .e-tbar-btn-text {
    font-variant: small-caps;
}
@font-face {
    font-family: "Material_toolbar";
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tShMAAAEoAAAAVmNtYXDoMOjqAAACDAAAAHhnbHlmIuy
19QAAAswAACNMaGVhZA6okZMAAADQAAAANmhoZWEIUQQkAAAArAAAACRobXR4jAAAAAAAAYAAAAC
MbG9jYYc0kUIAAAKEAAAASG1heHABOwG8AAABCAAAACBuYW1lx/RZbQAAJhgAAAKRcG9zdJZeEVU
AACisAAACGAABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAIwABAAAAAQAAAQsu/F8
PPPUACwQAAAAAANXLJlEAAAAA1csmUQAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAjAbAADgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnIQQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQ
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAA
AAAQAAAAEAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAGQAAAAEAAQAAQAA5yH//wA
A5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcACAAJAAoACwAMAA0ADgAPABAAEQASABMAFAAVABY
AFwAYABkAGgAbABwAHQAeAB8AIAAhACIAAAAAADIAjgFwAfgCIAKYAxIDSAO2BRYFMAVcBnIGugb
2ByoHQgguCNYJRgn6CiQKiAquCsgMFgzADOYNzg7WDvAQyBEyEaYABwAAAAAD9APzAAMABwAKAA4
AEgAVABkAADchNSElITUhJTkBBSE1ITUhNSEFFxEnITUhDAPo/BgBtgIy/c7+SgG2AjL9zgIy/c7
+Svr6A+j8GAxefV67Pl19Xvr6AfScXgAAAAIAAAAAA/QD9AAEAgAACUhNxc3AREfDyE/DxEvDyE
PDgOF/PbDisP9gQEBAwQEBgYICAgJCgoLCwsDCgsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQo
KCwsL/PYLCwsKCgkICAgGBgQEAwGz+qf6AYX89gsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQo
KCwsLAwoLCwsKCgkICAgGBgQEAwEBAQEDBAQGBggICAkKCgsLAAACAAAAAAPzA/QAQAC/AAABFQ8
PLw8/Dx8OAQ8ELwErAQ8FFR8FBxcPAxUfBzsBNx8LOwI/Cx8BOwE/Bj0BLwQ/Aic/BC8HKwEHLws
rAg8FArIBAgUGBwkKDAwODxAQERITEhIREQ8PDgwMCgkHBgUCAQECBQYHCQoMDA4PDxEREhITEhE
QEA8ODAwKCQcGBQL+zxUWFhUWfwUFBAUDBANqAgEBAgIDbgMDbwMCAQEBAmkDBAQEBQSEFBYWFxQ
CAgIDBAQEBcwFBAQEAwICAhQXFRYVgAQFBQQEAwNoAgEBAgIDcAEBAQNvAgIBAQEBA2gDBAQFBAW
DFBYWFxIBAgMDAwQFBcwFBAQDBAICAgAJCRIQEBAODgwLCgkHBgQDAQEDBAYHCQoLDA4OEBAQEhI
SEhAQEA4ODAsKCQcGBAMBAQMEBgcJCgsMDg4QEBASAc6ECwwNDjIBAQICA7QEBQQFBAMEUjIyVgM
EBAQFBAWwAwICATMODQwLhAQEBAMCAgECAgIDBAMEhAsMDQ4yAQECAgOwBAQFBAUEAwRSDAwaMlY
DBAQEBQQFsAMCAgEzDg0MC4QEAwQDAgICAgICAwQDAAAAAAMAAAAAA/MD2AAyADUAaQAAJRUfDTs
BPw41LwgPBwMhAScXAQ8GHQEfBQEfBjsBPwYBPwYvBwEDFgIDBAQGBgcICQkKCgsLCwsLCwoKCQg
ICAYGBAQDAQEDBAcMCQoLFCMtFQoJCQcFBHv96gEL04X+4gYFBAQCAgICAgIEBAUBNwcHBwgHCAc
IBwgHCAgHBwYBOAUEAwMCAQEBAQIDAwQFBv4PlwsLCwoKCQkIBwYGBAQDAgIDBAQGBgcICQkKCgs
LCwcPEBAYEBAPHCk3HRAQEBAQEAEIAQrThf7iBgcIBwgHCAgICAgHBwcH/skGBQQEAwIBAQIDBAQ
FBgE3BwcHBwgICAgHCAgHCAcGAfEABQAAAAAD9APzAAMABwALAA8AEwAANyE1ITchNSEnITUhNyE
1ISchNSEMA+j8GN4CLP3U3gPo/BjeAiz91N4D6PwYDF6AW5xefVqAXgAAAAAEAAAAAAP0A/QACQA
TABcAWwAAAQcVMzcXMzUjLwEjFTMbATM1IwElESERBxEfDyE/DxEvDyEPDgFro8ObnnROxOp0nZv
qTij+8AGW/NReAQEDBAQGBggICAkKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJCgo
LCwv89gsLCwoKCQgICAYGBAQDAQENAyOWliO4BSUBK/7VJQFSffzUAywR/PYLCwsKCgkICAgGBgQ
EAwEBAQEDBAQGBggICAkKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJCgoLCwAAAAA
CAAAAAAOWA/QAAwBpAAA3ITUhExUfHTsBPx01ESMRDw8vDxEjagMs/NRKAgIDAwUFBgcHCAkJCgs
LCwwNDQ0ODw4PEA8QERARERERERBEQDxAPDg8ODQ0NDAsLCwoJCQgHBwYFBQMDAgKLAQMFBggKCww
ODxARERMTFBQTExEREA8ODAsFCQcGBAKLDH0BsBERERERQEA8QDg8ODg0MDQsLCwoJCQgHBwYFBQQ
DAgEBAgMEBQUGBggICQkKCgsMDAwNDg4ODw8PEBAQERERAb7+RRQTEhIREA8NDQsKCAYFAwEBAwU
GCAoLDQ0PCBASEhMTAcUABQAAAAAD9APXAAIABQANABcAGgAAJTcjASM3ATM3MxczAyMFIQEVITU
hATUhJTMnAgJx4wG/vl/+/Fot+i1a3FD9RgEg/t4Bov7UAST+aAF/6XQobQET//47eHgCM07+XD5
NAaQ/UHMAAAAAAQAAAAAD9ALoAF8AABMhJz8PHxo3Lx8PDycMAbWyDQ0ODg8PDxAQEBERERIREhA
QEBAQDw8PDw4ODg0NDQwMFxYTEhAHBgYGBXUHBwgJCQoLCwwNDQ0PDg8QEBERERITEhMUExQVFBU
VFRgYFxcXFhYVFhQUFBMTEhGwARi6CwsJCggICAYGBgQEAwIBAQEBAgIDBAQFBQYHBwgICQkKFRY
```

YGhsODg8PDygUFBMTEhISERARDw8PDg0NDAsLCgoICAgGBgQEAwMBAQECAwQFBgcJCQoLDA0ODg+
6AAYAAAAAA/MD9AA/AGsAqwDrAO8BMwAAARUfDTsBPw09AS8ODw4lHwk7AT8IPQEvByMnByMPByU
fDz8PLw8PDiUfDz8OPQEvDSsBDw0lESERBxEfDyE/DxEvDyEPDgHhAgMFBQYHCAkKCgsLDA0NDA0
MCwsKCgkIBwYFBQMCAgMFBQYHCAkKCgsLDA0MDQ0MCwsKCgkIBwYFBQMC/scBAQEFBwgKCwYGBwY
GBgwKCAcFAQEBAQUHCAoMBgYGBwYGCwoIBwUBAQHzAQECBAQEBgYGCAcICQkJCgoJCQgJBwgGBgY
EBAMDAQEBAQMDBAQGBgYIBwkICQkKCgkJCQgHCAYGBgQEBAIB/qgBAQMEBAYGBwgICQoKCgsLCws
LCgkJCQcHBwUFAwMCAgMDBQUHBwcJCQkKCwsLCwsKCgoJCAgHBgYEBAMBAlD81F4BAQMDBAUGBgc
HCAkJCQkKAyYKCQkJCQgHBwYGBQQEAgEBAQECBAQFBgYHBwgICQkJCvzaCgkJCQkIBwcGBgUEAwM
BAWQNDAwMCwoKCQgHBgUFAwICAwUFBgcICQoKCwwMDA0NDAwLCwsJCQgHBwUEAwIBAQIDBAUHBwg
JCQsLCwwMMQYGBgsKCQcEAgEBAgQHCQoLBgYGBwYGCwoJBgUCAQECBQYJCgsGBvMJCgkICAgHBwY
FBQQDAwEBAQEDAwQFBQYHBwgICAkKCQoJCQkICAcGBwUFBAMCAQEBAQIDBAUFBwYHCAgJCQkGCws
KCgoJCAgHBgYEBAMBAQEBAwQEBgYHCAgJCgoKCwsLCwsKCQkJBwcHBQUDAwICAwMFBQcHBwkJCQo
LC9/81AMsA/zaCgkJCQkIBwcGBgUEBAIBAQEBAgQEBQUHBwcICQkJCQoDJgoJCQkJCAcHBwUFBAQ
CAQEBAQIEBAUFBwcHCAkJCQkAAAAAAgAAAAADtQP0AAMACgAANyE1IQEjCQEjESFKA2z8lAEG7AG
cAZzs/qAMfQIK/mQBnAFhAAYAAAAAA/QD8wADAAcACwAPABIAFgAANyE1ISUhNSE1ITUhNSE1KQE
RNwMhNSEMA+j8GAG2AjL9zgIy/c4CMv3O/kr6+gPo/BgMXn1efV19Xv4M+gGWXgAFAAAAAAPzA/M
AJQBpAKgArADwAAABFT8bIw8GBR8PNSMvDT8CJw8OHw8TByMPDBc/AzsBHwUzHwYzLxcjJREhEQc
RHw8hPw8RLw8hDw4CKg8QDw8ODg4NDQwMDAwKCwoKCQgJDw0KCQQDAgLxBAUGBggJC/7WDQ0NDg4
PDw8PEA8QEBAQEAoKCQkJCQgIBgQEBAUDAQEDBKsKCQgIBwYGBQUDAwMBAQEBAQEDAwQEBQYHBwg
ICgoL9g4OHA4ODQ4NDg0NDQwNDKkLDA8HCAkJCAgICA4DEAUFBQMEAu4EBwkKDQ8REwoKCwsMDAw
NDQ4ODg4PDy8KAZP81F4BAQMDBAUGBgcHCAkJCQkKAyYKCQkJCQgHBwYGBQQEAgEBAQECBAQFBgY
HBwgJCQkJCvzaCgkJCQkIBwcGBgUEAwMBAZz0AgMDBAQFBQYHBwgICQkJCgsLCwsYGhsbDw4PDwo
KCQgIBwaUDAsLCgkIBwcGBQQEAgIBAfEBAwMEBQYIBQcGBw8PEA8ODa0NDQ0ODg4ODw8PDw8PEA8
PEA8PEA8ODw8ODg0ODQ0MAj8BBAMDBAQFBgcHBwkJCqoGBQQBAQICBAMJEAcHCAgJCh0dHRsaGRc
XCgoKCQgICAcGBgYEBQMDBj781AMsA/zaCgkJCQkIBwcGBgUEAwMBAQEBAgQEBQYGBwcICQkJCQo
DJgoJCQkJCAcHBgYFBAQCAQEBAQIEBAUFBwcHCAkJCQkAAgAAAAAD8wPrAB8AMwAAEw8HHww/BBU
hNSEBNwkBPwcvCDYKCAcGBQMCAQECAwUGBwgKrwgJCgoKCgkINQKQ/Xv+20EBPAGOCgkHBgUDAgE
BAgMFBgcJCtMBoQsMDQ0NDg4ODg0ODQ0NDAuvBgUDAQECBAY0I14BJEH+xAGRCwwMDQ0ODg4ODg4
NDQwMC9QABgAAAAAD9AP0AAMADwATAB0AIQAnAAAlITUhIzMVIxUzFSMVMzUjNyE1ISMzBxUzNSM
3NSM3ITUhJzMVMzUjAQYC7v0S+n0/P328vPoC7v0S+nh4vHh4vPoC7v0S+j4/fWpeID4gPvrbXXw
/P3w/u14gvPoAAAAABQAAAAAD9APbAAIABQANABcAGgAAJTcjAyM3ATM3MxczAyMFIQEVITUhATU
hJzMnAgJx4x6+Xv76Wi39LF3fTwFlAST+3AGk/tIBJP5mw+l0JXMBGP/+N3h4AjRQ/lo+TQGpPk5
zAAABAAAAAAOsA/QACwAAATMDIxUhNSMTMzUhAXGd88gCPJ3zyP3EAx79xNbWAjzWAAAGAAAAAAP
0A9QAAwBDAEcAhwCLAMsAACUhNSEHFR8OPw49AS8ODw4TITUhBxUfDTsBPw09AS8ODw4TITUhBxU
fDj8OPQEvDg8OAQYC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgYGBQQEAwMCAQECAwMEBAUGBgY
HBwgICAgIBwgGBwYFBQUEAwMBAvoC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgYGBQQEAwMCAQE
CAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAvoC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgY
GBQQEAwMCAQECAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAkpeLwgIBwcHBwYFBQUDBAICAQE
BAQICBAMFBQUGBwcHBwgICAgIBwcGBgYFBAQDAwIBAQEBAgMDBAQFBgYGBwcICAFgXS4ICAgHBwY
GBgUEBAMDAgEBAgMDBAQFBgYGBwcICAgICAcHBwcGBQUFAwQCAgEBAQECAgQDBQUFBgcHBwcIAUB
dLggICAcHBgYGBQQEAwMCAQEBAQIDAwQEBQYGBgcHCAgICAgHBwcHBgUFBQMEAgIBAQEBAgIEAwU
FBQYHBwcHCAAAAwAAAADmQP0AAcAKACNAAABFSE1MxEhESUHFQ8GLwc/Bx8GJysBDw0VERUfDTM
hMz8NNRE1Lw0rAS8OKwEPDQEdAcZb/YQBbAEDBAYHBwkJCQkHBwYEAwEBAwQGBwcJCQkJBwcGBAO
svwkJCQgICAcGBgYEBAMCAgICAwQEBgYGBwgICAkCAgJCQn9KQkJCQgICAcGBgUFBAMCAgI
CAwQFBQYGBwgICAkJCQLXCQkJCAgIBwYGBgQEAwICCgkJCAgIBwYGBQQEAwICAgIDBAQFBgYHCAg
ICQkAAQAAAAAD9ALoAGAAAAExLw8PHxc/Gh8PByERA0QREhMTFBQUFRYWFhcXFxgYFRUVFBUUExQ
TEhMSEREREBAPDg8NDQ0MCwsKCQkIBwd1BQYGBgcQEhMWFwwMDQ0ODw8PDxAQEBAQEhESEREQ
QERAQDw8PDg4NDbIBtQIuDw4ODQwLCgkJBwYFBAMCAQEBAwMEBAYGCAgICgoLCwwNDQ4PDw8REBE
SEhITExQUKA8PDw4OGxoYFhUKCQkICAcHBgUFBAQDAgIBAQEBAgMEBAYGBggICAoJCwu6AdAAAAA
OAAAAAAP0A/MAAgAFAAgACwAQABQAFwAbAB4AIQApAC0AMQB1AAABETclFzUXNyMFNyETFQUhEQE
hJRMlMycFMSEnBzcnBxcRBRMDBSUDEyc0BEQMlIwUDEQcRHw8hPw8RLw8hDw4CGcj+ZaG3MJb+wM7
+4jQBCv6EAy7+ggEKdP1S3JkBCwEjWemWlvrIATJ0dP7n/up3dwEWAZhy/vQ0/vZyXgEBAwQEBgY
ICAgJCgoLCwsDCgsLCwoJCAgHBwUFBAMBAQECAwQEBQUGBwcICQkKCwsL/PYLCwsKCgkICAgGBQQ
EAwEBxv7fWSQ730Bky8v+9QNxAYH+f28BHx2ZmcukmTgJywEeP/7n/ud3dwEZARl3Bv5xAR1ycv7
yAYAR/PYLCwsKCgkICAgHBwUFBAMBAQEDBAUFBwcICQkKCwsLCwoJCAgHBwUFBAMBAQEDBAUFBwcI

```
BAwQEBgYIBwkJCgoLCwAAAAAFAAAAAAP0A/MAAwAHAAsADwATAAA3ITUhJSE1ISUhNSElITUhJSE
1IQwD6PwYAVgCkP1w/qgD6PwYAVgCkP1w/qgD6PwYDF6AW5xefV19XgAAAAAKAAAAAAP0A/MAAwA
HAAsADwATABcAGwAfACMARwAAARUjNSMVIzUjFSM1ARUjNSMVIzUjFSM1JRUjNSMVIzUjFSM1JxE
fByE/BxEvByEPBgOW+j7bP9oDLPo+2z/aAyz6Pts/2l4BAwUGAwgJCgOJCgkJBwYDBAIBAwUGAwg
JCvx3CgkJBwYFAwElvb27u7u7ARrb29vb29v6vLy8vLy8hvyCCwoJBwQGBAIBAwUHBwUJCgOECwo
JBwQGBAIBAwUGCAkKAAAAAAUAAAAAA/QD8wADAAcACwAPABMAADchNSE1ITUhNSE1ITUhNSE1ITU
hDAPo/BgCkP1wA+j8GAKQ/XAD6PwYDF6BV59efVqAXgAAAAADAAAAAAP0A00AAwAHAAsAADchNSE
1ITUhNSE1IQwD6PwYA+j8GAPo/Bizb6Zwpm8AAAAABQAAAAD9AP0AD8AXwCfAKQBIgAAJQ8PLw8
/Dx8OExUPBSsBLwU9AT8FOwEfBQMPDy8PPw8fEAE1IwUVHw8zPwMXBy8FDw8fDz8PNS8DNwEzNQE
/BS8PDw4BOAEBAwMEBQYGBwgICQkKCgoKCgoJCQgIBwYGBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgo
KCgkJCAgHBgYFBAMDAeICAgMDBQUFBQUFAwMCAgICAwMFBQUFBQUDAwIC4QEBAwMEBQYGBwgICQk
KCgoKCgoJCQgIBwYGBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAgHBgYFBAMDAftkAV6W/K4
BAwUHCAoMDQ4PERETExQUCwsVFBN2dgkKCgoVFhQUExMREQ8ODQwKCAcFAwEBAwUHCAoMDQ4PERE
TExQUFBQTExERDw4NDAoIBwUDAQEEBgd2AV6W/ZYFBAMCAwEBAwUHCAoMDQ4PERETExQUFBQTExE
RDw4NDAoIBwUD1AoKCgkJCAgHBgYFBAMDAQEBAQMDBAUGBgcICAkJCgoKCgoKCQkICAcGBgUEAwM
BAQEBAwMEBQYGBwgICQkKCgEiBQUFAwMCAgICAwMFBQUFBQUDAwICAgIDAwUFAScKCgoJCQgIBwY
GBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAgHBgYFBAMDAQEBAQMDBAUGBgcICAkJCgqgZAF
eMpYKChQTExERDw4NDAoIBwUDAQEEBgd2dgUEAwIDAQEDBQcICgwNDg8RERMTFBQUFBMTEREPDg0
MCggHBQMBAQMFBwgKDA0ODxEREXMUFAsLFRQTdv6iMgJqCQoKChUWFBQTExERDw4NDAoIBwUDAQE
DBQcICgwNDg8RERMTFAADAAAAAANXA7UAIgBFAJMAAAEzHw4PDisBNRMzHw4PDisBNQMhPxEvDz8
PLxghAkgKCgkJCAgHBwYGBAQEAgEBAQEDAwQFBgYHBwgJCAkKCeDACgoJCQgIBwcGBgQEBAIBAQE
BAgQEBAYGBwcICAkJCgrAwAHDDQwMDBcWFRMSEQ8NDAoHBgQBAQIDBAYHBwkKCgsNDA4ODwsLCgo
KCAgIBgYFBQMDAQEBAQECAwQEBAUGDA8QEhQVFgwMDA0NDQ0N/nABogICAwQEBgYGBwgICQkKCQo
KCQgJBwgGBgUFBAMCArsBdwICAwQEBgYGCAcICQkKCQoKCQkIBwgGBgYEBAMCArv9MQEBAQIGCAo
MDg8REhQUFhcYGBERERAQEA4ODgwMDAoJCQcICQkKCgoLDAsMDAwMDQwNDQwNDQwMCwwLCxQUERA
ODQoFAwQDAgEBAQAABQAAAAD9APzAAMABwALAA8AEwAANyE1ITUhNSE1ITUhNSE1ITUhNSEMA+j
8GAPo/BgD6PwYA+j8GAPo/BgMXn1enF59XX1eAAAAAAEAAAAAA9QD1ADUAAATHx8/DxcRIRcPDy8
fPx8fDzMvHw8eKwECAwQFBggICQoMDA0ODhAQERISExQUFRUWFhcXGBgYGBgXFxcWFhUVFBQTEhI
REIr+ZrsMDA0ODg4PEBAQEBESERISEhIREhEQEQ8QDw8ODg0NDAwLCgoJCQgHBgYEBAQCAQEBAQI
EBAQGBgcICQkKCgsMDA0NDg4PDxAPERAREhESEhwcGxoaGBgWFRQSEQ8OCwp7BQYHCAgJCQoLCww
NDQ4ODg8QEBERERISEhMTFBMUFRQYGBgXFxYWFRUUFBMSEhEQEA4ODQwMCgkICAcGBggICgkLCww
MDQ0ODg8PDxAREBESERISEhIREhEQERAPDw8ODg0NDAwLCgoJCAcHBggICgkLCwwMDQ0ODg8PDxA
REBESERISEhIREhEQERAPDw8ODg0NDAsLCgoJCAcHBggICgkLCwsNDA4ODg8QEBARERISEhMTExE
REBEQEBAQDw8ODg4ODA0OAR1W/uMuDgoKCQkIBwYGBgQEAwMCAQICAwQFBgcHCAkKCgsMDA0NDg8
PDxARERERER
```

```
BAgIEBAUFBgYHBwgICQgJCQgHCAYHBgUFBAMDAgEBAQECAwMEBQUGBwYIBwgJCQgJCAgHBwYGBQU
EBAICAQEBAQICBAQFBQYGBwcICAn+xhoZGRgZFxcXFhYVFBQTEhERDw8ODQwLCwkHBwYEAwIBAgI
CBAQFBQYGBwcICAkICAgIBwcGBgsGBwYIBwgICQkIBwgGBwYFBQQDAwIBAQECAgMEBQUFBgcHCAg
JCQoKCwoMCwwNDA0NDg0ODg4XFxYWFRUVFBQTExIRERAPDw4NDQsLCgkIBwYFBAMBAQECAwQGBwc
JCwsMDQ4PDxEREhMUFBUWFhcXFxkYGRkAAgAAAAAD9AO1AAgAVAAAARchFSEHFzcnJREVHw4hPw4
9ASMVIREhFTM9AS8OIQ8OAtV1/k0BsHI/4OD8+AICAwQFBQYHBwcICQkJCQHPCQkJCQgHBwcGBQU
EAwICXP4xAc9cAgIDBAUFBgcHBwgJCQkJ/jEJCQkJCAcHBwYFBQQDAgICoHRYdD7e3oD9RAkJCAg
IBwcGBgUEBAMCAQEBAQIDBAQFBgYHBwgICAkJzMwCvMzMCQkICAgHBwYGBQQEAwIBAQEBAgMEBAU
GBgcHCAgICQADAAAAAAOvA/QAAwBHAF0AAAERIREHERUfDTMhMz8OES8OIyEjDw0nETMRITUhIw8
NA1X+DFsCAgMEBQUGBgcICAgJCQkB9AkJCQgICAcGBgUFBAMCAQEBAQIDBAUFBgYHCAgICQkJ/gw
JCQkICAgHBgYFBQQDAgK2WQIT/e0JCQkIBwgHBgYFBAQDAgEC4/2EAnwF/YgJCQgJCAcHBgYGBAQ
DAgICAgMEBAYGBgcHCAkICQkCeAkJCQgICAcGBgUFAwMDAQEDAwMFBQYGBwgICAkJsv2EAnxbAgI
DBAUFBgYHCAgICQkAAAASAN4AAQAAAAAAAAABAAAAAQAAAAAAAQAQAAEAAQAAAAAAgAHABEAAQA
AAAAAAwAQABgAAQAAAAAABAAQACgAAQAAAAAABQALADgAAQAAAAAABgAQAEMAAQAAAAAACgAsAFM
AAQAAAAAACwASAH8AAwABBAkAAAACAJEAAwABBAkAAQAgAJMAAwABBAkAAgAOALMAAwABBAkAAwA
gAMEAAwABBAkABAAgAOEAAwABBAkABQAWAQEAAwABBAkABgAgARcAAwABBAkACgBYATcAAwABBAk
ACwAkAY8gdG9vbGJhci1tYXRlcmlhbFJlZ3VsYXJ0b29sYmFyLW1hdGVyaWFsdG9vbGJhci1tYXR
lcmlhbFlcnNpb24gMS4wdG9vbGJhci1tYXRlcmlhbEZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAdABvAG8AbABiAGEAcgAtAG0
AYQB0AGUAcgBpAGEAbABSAGUAZwB1AGwAYQByAHQAbwBvAGwAYgBhAHIALQBtAGEAdABlAHIAaQB
hAGwAdABvAG8AbABiAGEAcgAtAG0AYQB0AGUAcgBpAGEAbABWAGUAcgBzAGkAbwBuACAAMQAuADA
AdABvAG8AbABiAGEAcgAtAG0AYQB0AGUAcgBpAGEAbABGAG8AbgB0ACAAZwBlAG4AZQByAGEAdAB
lAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHU
AZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAAAAAAKAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAjAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4BDwEQAREBEgE
TARQBFQEWARcBGAEZARoBGwEcAR0BHgEfASABIQEiASMBJAAQVGV4dF9PdXRkZW50XzAwMQtQaWN
0dXJlXzAwMQxTZXR0aW5nc18wMDEQQ29sb3JfcGlja2VyXzAwMhBBbGlnbl9DZW50ZXJfMDA2CEx
pbmVfMDAxDVVuZGVybGluZV8wMDEMU29ydF9aLUFfMDAxCFVuZG9fMDAxEENoYXJ0X2J1YmJsZV8
wMDELRG93bmxvYWRfMDAPVGV4dF9pbmRlbnRfMDAxEkNoYXJ0X0RvdWdobnV0XzAwMQlDbGVhcl8
wMDINTnVtYmVyaW5nXzAwMQxTb3J0X0EtWl8wMDEKSXRhbGljXzAwMQtCdWxsZXRzXzAwMQlQYXN
0ZV8wMDEIUmVkb18wMDEPQ2hhcnRfcmFkYXJfMDAxD0FsaWduX1JpZ2h0XzAwMQlUYWJsZV8wMDE
OQWxpZ25fTGVmdF8wMDEITWVudV8wMDEHQ3V0XzAwMghCb2xkXzAwMRFBbGlnbl9KdXN0aWZ5XzA
wMQpSZWxvYWRfMDAxClNlYXJjaF8wMDEKVXBsb2FkXzAwMQpEZXNpZ25fMDA1CkV4cG9ydF8wMDE
IQ29weV8wMDIAAA==)
      format("truetype");
    font-weight: normal;
    font-style: normal;
}
 .e-cut-icon:before {
    content: "\e719";
  }

  .e-copy-icon:before {
    content: "\e721";
  }

  .e-paste-icon:before {
    content: "\e712";
  }

  .e-color-icon:before {
    content: "\e703";
  }

  .e-bold-icon:before {
    content: "\e71a";
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  .e-underline-icon:before {
    content: "\e706";
  }
  .e-italic-icon:before {
    content: "\e710";
  }
  .e-ascending-icon:before {
    content: "\e70f";
  }

  .e-descending-icon:before {
    content: "\e707";
  }

  .e-clear-icon:before {
    content: "\e70d";
  }

</style>
```

{% previewsample "page.domainurl/code-snippet/toolbar/responsive-mode/textmode-cs1" %}

Accessibility in Vue Toolbar component

The Vue Toolbar component has been designed, keeping in mind the WAI-ARIA specifications, and applying the WAI-ARIA roles, states, and properties along with keyboard support for people who use assistive devices. WAI-ARIA accessibility support is achieved through attributes like aria-label, and aria-orientation, It provides information about elements in a document for assistive technology. The component implements keyboard navigation support by following the WAI-ARIA practices, and has been tested in major screen readers.

The accessibility compliance for the Toolbar component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

ARIA attributes

Toolbar component is designed by considering WAI-ARIA standard. Toolbar is supported with ARIA Accessibility which is accessible by on-screen readers, and other assistive technology devices. The following list of attributes are added in the Toolbar.

| **Property** | **Functionalities** |

| --- | --- |

| role="toolbar" | Attribute is set to the ToolBar element describes the actual role of the element. |

| aria-orientation | Attribute is set to the ToolBar element to indicates the ToolBar orientation. Default value is `horizontal`. |

| aria-label | Attribute is set to ToolBar element describes the purpose of the set of toolbar. |

| aria-expanded | Attribute is set to the ToolBar Popup element to indicates the expanded state of the popup.|

| aria-haspopup | Attribute is set to the popup element to indicates the popup mode of the Toolbar. Default value is false. When popup mode is enabled, attribute value has to be changed to `true`. |

| aria-disabled | Attribute set to the ToolBar element to indicates the disabled state of the ToolBar. |

Keyboard interaction

Keyboard navigation is enabled by default. Possible keys are

| Key | Description |

|----------|---|

| Left | Focuses the previous element. |

| Right | Focuses the next element. |

| Enter | When focused on a ToolBar command, clicking the key triggers the click of Toolbar element. When popup drop-down icon is focused, the popup opens. |

| Esc(Escape) | Closes popup. |

| Down | Focuses the next popup element. |

| Up | Focuses the previous popup element. |

| Home | Moves focus to the first Toolbar. |

| End | Moves focus to the last Toolbar. |

| Tab | To Move focus through the interactive elements. |

| Shift + Tab | To Move focus through the interactive elements. |

Ensuring accessibility

The Toolbar component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Toolbar component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Toolbar component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/toolbar.html" %}

See also

- Accessibility in Syncfusion Vue components

Style in Vue Toolbar component

The following content provides the exact CSS structure that can be used to modify the control’s appearance based on user preference.

Customizing Toolbar

Use the following CSS to customize the Toolbar.

```css
.e-toolbar {
border: 5px solid rgb(173, 255, 47);
}
```

Customizing the Toolbar items

Use the following CSS to customize the items of Toolbar.

```css
.e-toolbar .e-toolbar-item {
background: #add8e6;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
border: 1px solid #5a70cc;
}
```

Use the following CSS to customize the button in the items of Toolbar.

```css
.e-toolbar .e-tbar-btn {
background: #add8e6;
border: 1px solid #5a70cc;
}
```

Customizing Toolbar's item icon

Use the following CSS to customize the item icon of Toolbar control.

```css
.e-toolbar .e-tbar-btn .e-icons {
background: #185655;
color: #d7f9d4;
}
```

Customizing the hover state of Toolbar control

Use the following CSS to customize the toolbar item when hovering.

```css
.e-toolbar .e-tbar-btn:hover {
background: #c0e3a1;
border: 1px solid green;
}
```

Customizing selected item of Toolbar control

Use the following CSS to customize the selected toolbar item.

```css
.e-toolbar .e-tbar-btn:focus {
background: #add8e6;
border: 1px solid #5a70cc;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

How To

Set command customization in Vue Toolbar component

The htmlAttributes property of the Toolbar item is used to set the HTML attributes ('ID', 'class', 'style' ,'role') for the commands.

When style attributes are added, if the same attributes were already present, they will be replaced. This is not so in the case of class

attribute. Classes will be added to the element instead of replacing the existing ones.

Single or multiple CSS classes can be added to the Toolbar commands using the Toolbar item cssClass property.

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:100%;">
        <br>
         <ejs-toolbar >
           <e-items>
             <e-item text='Bold'  :htmlAttributes = 'boldAttribute'></e-
item>
             <e-item text='Italic' :htmlAttributes = 'italicAttribute'></e-
item>
             <e-item text='Underline' :htmlAttributes =
'underAttribute'></e-item>
             <e-item type='Separator'></e-item>
             <e-item text='Uppercase' cssClass = 'e-txt-casing'></e-item>
          </e-items>
    </ejs-toolbar>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(ToolbarPlugin);
export default {
name: 'app',
   data() {
  return {
      boldAttribute: { 'class': 'custom_bold', 'id': 'itemId' }
      italicAttribute :{ 'class': 'custom_italic' }
      underAttribute : { 'class': 'custom_underline' }
   }
   }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
.custom_bold .e-tbar-btn-text {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
     font-weight: 900;
}
.custom_italic .e-tbar-btn-text {
   font-style: italic;
}
.custom_underline .e-tbar-btn-text {
   text-decoration: underline red;
}
.e-txt-casing .e-tbar-btn-text {
   font-variant: small-caps;
}
.e-undo-icon:before {
    content: "\e76e"
}
.e-redo-icon:before {
    content: "\e726"
}
.e-cut-icon:before {
    content: "\e759"
}
.e-copy-icon:before {
    content: "\e70a"
}
.e-paste-icon:before {
    content: "\e77e"
}
.e-color-icon:before {
    content: "\e778"
}
.e-bold-icon:before {
    content: "\e71c"
}
.e-underline-icon:before {
    content: "\e75a"
}
.e-alignleft-icon:before {
    content: "\e75e"
}
.e-alignright-icon:before {
    content: "\e75f"
}
.e-aligncenter-icon:before {
    content: "\e74b"
}
.e-alignjustify-icon:before {
    content: "\e756"
}
.e-upload-icon:before {
    content: "\e725"
}
.e-download-icon:before {
    content: "\e824"
}
.e-indent-icon:before {
    content: "\e742"
}
.e-outdent-icon:before {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    content: "\e746"
}
.e-clear-icon:before {
    content: "\e760"
}
.e-reload-icon:before {
    content: "\e837"
}
.e-export-icon:before {
    content: "\e7fb"
}
.e-italic-icon:before {
    content: "\e709"
}
.e-bullets-icon:before {
    content: "\e75c"
}
.e-numbering-icon:before {
    content: "\e733"
}
.e-ascending-icon:before {
    content: "\e73f"
}
.e-descending-icon:before {
    content: "\e79d"
}
.e-zoomin-icon:before {
    content: "\e7d8"
}
.e-zoomout-icon:before {
    content: "\e7ea"
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toolbar/how-to/customization-cs1" %}

Set tool tip to the commands in Vue Toolbar component

The tooltipText property of the Toolbar item is used to set the HTML Tooltip to the commands that can be viewed as hint texts on mouse hover.

To change the tooltipText to ej2-tooltip component:

- Import the Tooltip module from ej2-popups,and initialize the Tooltip with the Toolbar target. Refer to the following code example:

APP.VUE

```
<template>
  <div id="app">
    <div id = 'Tooltip'  id ='Toolbar' :created="oncreated">
         <ejs-toolbar >
    <e-items>
             <e-item text='Cut'></e-item>
             <e-item text='Copy'></e-item>
             <e-item text='Paste'></e-item>
```

```
            <e-item type='Separator'></e-item>
            <e-item text='Bold'></e-item>
            <e-item text='Italic'></e-item>
            <e-item text='Underline'></e-item>
          </e-items>
    </ejs-toolbar>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
import { Tooltip } from "@syncfusion/ej2-popups";
Vue.use(ToolbarPlugin);
export default {
  name: 'app',
   data() {
    },
    methods:{
      oncreated: function(e){
      var tooltip  = new Tooltip({ target: '#Toolbar',content: 'This is
Toolbar'});
       tooltip.appendTo('#Tooltip');
      }
    }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/toolbar/how-to/Tooltip-cs1" %}

Set item wise custom template in Vue Toolbar component

The Toolbar supports adding template commands using the template property. Template property can be given as the HTML element that is

either a string or a query selector.

As a string

The HTML element tag can be given as a string for the template property. Here, the checkbox is rendered as a HTML template.

`html

template: " <input type='checkbox' id='check1' checked=''>Accept</input> "

`

As a selector

The template property also allows getting template content through query selector. Here, checkbox 'ID' attribute is specified in the template.

Copyright © 2001 -2024 Syncfusion Inc.

`ts

template: "#Template"

`

APP.VUE

```
<template>
  <div id="app">
    <div id='container' style="margin:50px auto 0; width:100%;">
        <br>
         <ejs-toolbar >
    <e-items>
             <e-item text='Cut'></e-item>
             <e-item type='Separator'></e-item>
             <e-item text='Paste'></e-item>
             <e-item type='Separator'></e-item>
             <e-item :template='templateEle'></e-item>
             <e-item text='Undo'></e-item>
             <e-item text='Redo'></e-item>
             <e-item :template='templateEleId'></e-item>
          </e-items>
    </ejs-toolbar>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(ToolbarPlugin);
export default {
  name: 'app',
  data () {
  return {
    templateEle: '<input placeholder="Search" style="height:27px;"/>';
    templateEleId: '#Template';
}
}
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/toolbar/how-to/toolbar-items-templateID-cs1" %}

Add toggle button in Vue Toolbar component

Toolbar supports to add a toggle Button by using the template property. Refer below steps

- By using Toolbar template property, pass required HTML String to render toggle button.

Copyright © 2001 -2024 Syncfusion Inc.

`

<e-item template='<button class="e-btn" id="media_btn"></button>'></e-item>

`

- Now render the toggle Button into the targeted element in Toolbar created event handler and bind click event for it. On clicking the

toggle Button, change the required icon and content based on current active state.

APP.VUE

```
<template>
  <div id="app">
        <br>
         <ejs-toolbar id="toolbar" class="toggle" :created='create'>
    <e-items>
              <e-item template='<button class="e-btn"
id="media_btn"></button>'></e-item>
                <e-item type='Separator'></e-item>
                <e-item template='<button class="e-btn"
id="zoom_btn"></button>'></e-item>
                <e-item type='Separator'></e-item>
                <e-item template='<button class="e-btn"
id="undo_btn"></button>'></e-item>
                <e-item type='Separator'></e-item>
                <e-item template='<button class="e-btn"
id="filter_btn"></button>'></e-item>
                <e-item type='Separator'></e-item>
                <e-item template='<button class="e-btn"
id="visible_btn"></button>'></e-item>
          </e-items>
    </ejs-toolbar>
    <br/>
        <div id="content">
          This content will be hidden, when you click on hide button and
toggle get an active state as show, otherwise it will be visible.
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
import { Button} from "@syncfusion/ej2-buttons";
Vue.use(ToolbarPlugin);
export default {
  name: 'app',
  data () {
},
  methods: {
    create: function(args) {
     var zoomBtn = new Button({ cssClass: `e-flat`, iconCss: 'e-icons e-
zoomin-icon', isToggle: true });
    zoomBtn.appendTo('#zoom_btn');
    var mediaBtn = new Button({ cssClass: `e-flat`, iconCss: 'e-icons e-
play-icon', isToggle: true });
```

```
    mediaBtn.appendTo('#media_btn');
    var undoBtn = new Button({ cssClass: `e-flat`, iconCss: 'e-icons e-undo-
icon', isToggle: true });
    undoBtn.appendTo('#undo_btn');
    var filterBtn = new Button({ cssClass: `e-flat`, iconCss: 'e-icons e-
filter-icon', isToggle: true });
    filterBtn.appendTo('#filter_btn');
    var visibleBtn = new Button({ cssClass: `e-flat`, iconCss: 'e-icons e-
hide-icon', isToggle: true, content:'Hide'});
    visibleBtn.appendTo('#visible_btn');
    //Toggle button click event handlers
    zoomBtn.element.onclick = (): void => {
        if (zoomBtn.element.classList.contains('e-active')) {
            zoomBtn.iconCss = 'e-icons e-zoomout-icon';
        } else {
            zoomBtn.iconCss = 'e-icons e-zoomin-icon';
        }
    };
    mediaBtn.element.onclick = (): void => {
        if (mediaBtn.element.classList.contains('e-active')) {
            mediaBtn.iconCss = 'e-icons e-pause-icon';
        } else {
            mediaBtn.iconCss = 'e-icons e-play-icon';
        }
    };
    undoBtn.element.onclick = (): void => {
        if (undoBtn.element.classList.contains('e-active')) {
            undoBtn.iconCss = 'e-icons e-redo-icon';
        } else {
            undoBtn.iconCss = 'e-icons e-undo-icon';
        }
    };
    filterBtn.element.onclick = (): void => {
        if (filterBtn.element.classList.contains('e-active')) {
            filterBtn.iconCss = 'e-icons e-filternone-icon';
        } else {
            filterBtn.iconCss = 'e-icons e-filter-icon';
        }
    };
    visibleBtn.element.onclick = (): void => {
        if (visibleBtn.element.classList.contains('e-active')) {
            document.getElementById('content').style.display = 'none';
            visibleBtn.content = 'Show';
            visibleBtn.iconCss = 'e-icons e-show-icon';
        } else {
            document.getElementById('content').style.display = 'block';
            visibleBtn.content = 'Hide';
            visibleBtn.iconCss = 'e-icons e-hide-icon';
        }
    };
}
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
        @font-face {
            font-family: 'toolbar';
            src:
                url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMv1tCfsAAAEoAAAAVmNtYXCnKKeOAAABrAAAAEhnbHlm5Ce
ZPwAAAgwAAApsaGVhZBKvqTUAAADQAAAANmhoZWEIUQQMAAAArAAAACRobXR4LAAAAAAAAYAAAAA
sbG9jYRGwFFwAAAH0AAAAGG1heHABGgFNAAABCAAAACBuYW1lQozdSwAADHgAAAIlcG9zdFKJfTQ
AAA6gAAAAlAABAAAEAAAAAFwEAAAAAAAD9AABAAAAAAAAAAAAAAAAAAAACwABAAAAAQAAtluoWF8
PPPUACwQAAAAAANfOsiIAAAAA186yIgAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAALAUEABQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABApwCnCQQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAAFAA
EADQAAAAEAAQAAQAApwn//wAApwD//wAAAAEABAAAAAEAAgADAAQABQAGAAcACAAJAAoAAAAAATw
CsAMyA6wDugPQBGQE+gUkBTYABAAAAAAD8wMoAEAAgQDHARwAAAE7AR8ODw8vDjU/DgcdAR8OPw8
1Lw4PDjcfEA8OKwEvDj8SMycPEhUfEz8TNS8TIw8BAf8BDg4NDQ0LCwsJCAgFBQMCAQECAwUGBwg
JCgsLDQ0NDg8PDg0NDQwLCgoIBwYEAwEBAwQFBgcICQoLCwwMDQ28AwQGCAoMDQ8RERITFBQVFRQ
UExIQEA4NDAoIBwUDAQMFBwgLDA0PEBESEhMUExcTExIREBAODQwKCQcGBOYODh0cHR0dHR8WFhY
VFRUUFB0eHx8fISAiFhcWFhUWFhUbGBgXGRgYGSAfHh4dHR0cGBkYGRoaGxwUExQUExQTFBITEhJ
VFhYWFhYWFhcfHh4dHRwbGwYEAgEDAwQfICEhIiIkJBscGxsbGxsbIAwZGhkZGRogJSUkIyIiIRM
FAwICAwUZGBcYGBkZGSIhICEgICEgHBUWFQKMAgQEBgcICQsLCw0NDQ4ODw4NDQ0LCwoJCAcGBQM
CAQECBAQHBwkJCwwNDQ0ODw4ODQ0MDAsKCggIBgYEBAKBCwsUFBQTEhEQDgwLCQcFAwEBAwUGCQo
LDQ8PERITExUVFBQTExIREA8NDAoJBgUDAQEDBQYICgsMDg8QERETE8sBAgYHCgwOEBMPEBASExQ
UFh8dGxkXFRMRCggIBgQEAgIEBQcJCgwQExQXGBkcHhoYFxUUExIRCwoJCAcGBQQDAgE4BQYHBwk
KDAwTFBYYGRocHggIBwgGBwYFIyAeHBkYFRMNDAkIBgQDAQEBAwQGCAkLDxQXGBwdICMWBwcICAc
IBx0YFxUVExISFBIQDQsIBwMCAgIAAAAFAAAAAAP0A9QAIAA+AIIAxgFAAAABDw8jLwY3HwYnMx8
DBy8FPQE/DiUfBw8PLwc3Hwc/DzUvBjcXJx8EBy8GKwEPDhUfBgcvBz8SMyUHLwgjDxQVHwsPBB8
HPwQfBzM/EjUvDD8DPQEvBg8CAo0BAgQEBgcICQoLDAwNDQ8ODAsLCwoLCQnFBgYFBAMCAY0BERE
QD74EAwMCAgEBAwQFBgcICQoKDAwMDQ0BFBYWFhUVFRQUHR4fHx8gISIXFhcWFxYWFhUVFBUVFRY
VJw4PDxAQERISFRQUExEREA4NDAoIBwUDAQMDBQcICQsyGswPDh4dHSMNDA4NDg4PDhcTExIREBA
ODQwKCQcGBAIBAwMFBgcIORsaGhkZGRgZGBkYGhkaGxwUExQUExQTFBITEhIBe74UFBQUFBQUFBw
dHB0WFhYWFhYWFx8fHR4cHBwbBQQCAQMDBBoaGhsbHBwc2gQDAQEBAQMEBQUGBgYGBQXoDRwbGxs
bGhsaGhkaGRoaGiAlJSQjIiIhFAQDAgIDBRkYFxgYGRkZFRWvBAMCAgMEBQUGBgYGBQIQDw4ODQw
MCgoJCAcGBQMCAQICAwQFBQfFCQoKCwoMC4EBAwUHvQgJCAkKCQkKDg0NDAsLCwkJBwcGBAMCBw8
QERISFBUWHx0bGRcVEhELCAgGBAQBAQEBAwQFBwgJJwoKBwcFBAIBAQMFBwgKDA0OEBESEhQUFRI
SERAQEA8OMQ9GAgIFCAokCAYGBAQCAgMFBwgJCw0NDxAREhITFBAQEA8PDw4OOBARExQWFhkaGRg
XFhQTEhALCgkIBwYFBQMCAdK/CQcHBQUEAwIBAwQFBQcICQoLDBQUFhcZGxweCAcIBwcGBgYcGxk
YFxQUEtoFBgYFBgYGBQQDAQEBAgIE6AcLCwgGBQQBAQMFBgcJCxAUFhkbHiAjFQgHCAcIBwgdGBY
WFBQSEQ4MsAQGBgYGBQYFBAMBAQECAgAAAAACAAAAAAO+A/QACwBtAAABFTMVIxUjNSM1MzUnDxE
VHxk/BAE3AT8GNS8YDwoBxZ+faJ+fmQ0NDQsLCgkICAcGBQEAwMBAQECAgQEBQUHBwkJCQsKCgo
LFhcYGhkbGxscHBscGxoBCpP+8w8OCwkHBQMBBAYICg0OCQgODg4PEBARERISEhMTExMTHBwcGw4
NDQ0MDQwDbqNkn59noDsLDAwNDQ4ODg8PEBAPERAQERARERAREBEQEBAPEA8ODgwLCgoSEQ4MCQg
GAwEBAwYHC/6udAFUGBgZGhobGxwbGxsaGhkYDAwQDw0NDAsKCAgHBQUDAwEBAQQGCAUFBgcIBwk
AAAIAAAAAA74D9AADAGUAAAEVITU3DxEVHxk/BAE3AT8GNS8YDwoCZ/5XBg0NDQsLCgkICAcGBgQ
EAwMBAQECAgQEBQUHBwkJCQsKCgoLFhcYGhkbGxscHBscGxoBCpP+8w8OCwkHBQMBBAYICg0OCQg
ODg4PEBARERISEhMTExMTHBwcGw4NDQ0MDQwCy2dn3gsMDA0NDg4ODw8QEA8REBAREBEREBEQERA
QEA8QDw4ODAsKChIRDgwJCAYDAQEDBgcL/q50AVQYGBkaGhsbHBsbGxoaGRgMDBAPDQ0MCwoICAc
FBQMDAQEBBAYIBQUGBwgHCQABAAAAAAANhA/QAAgAANwkBngLE/TwMAfQB9AAAAgAAAAADxwP0AAM
ABwAAJSERIQEhESECaQFe/qL90AFe/qIMA+j8GAPoAAABAAAAAAP0AxUAgAAAAQ8HJwMhJz8XHx8
PBzM/Ay8eKwEPDQEQDg0MDAsKCQiKIQGhkAUHCAkLDQ4PCwsLDA0MDg0NDg4PDg8PDw8PDg8ODg4
ODQ0NDAwMCwsKCgkICAgGBgUFBAMCAgEBAQECAwMFBQaOBwYDAQEBAwMFBgYICAoKDAwNDg8QEBE
REhITExQUFBUVFRYVFhUWFBUVFBMTExMSEREQApYOEA8REBESEpr+U6EVFBQUExESEAoKCQgICAY
GBQUEAwICAQEBAQICAwQFBQYGCAgICQoKCwsMDAwNDQ0ODg4ODw8ODxAQDxAPDg8OHR4fHxUWFRU
VFBQUExMSEREQApYOEA8REBESEpr+U6EVFBQUExESEAoKCQgICAYGBQUEAwICAQEBAQICAwQFBQY
GCAgICQoKCwsMDAwNDQ0ODg4ODw8ODxAQDxAPDg8OHR4fHxUWFRUVFBQUExMSEREQApYOEA8REBE
SEpr+U6EVFBQUExESEAoKCQgICAYGBQUEAwICAQEBAQICAwQFBQYGCAgICQoKCwsMDAwNDQ0ODg4
ODw8ODxAQDxAPDg8OHR4fHxUWFRUVFBQUExMSEREQApYOEA8REBESEpr+U6EVFBQUExESEAoKCQg
ICAYGBQUEAwICAQEBAQICAwQFBQYGCAgICQoKCwsMDAwNDQ0ODg4ODw8ODxAQDxAPDg8OHR4fHxU
WFRUVFBQUExMSEREQApYOEA8REBESEpr+U6EVFBQUExESEAoKCQgICAYGBQUEAwICAQEBAQICAwQ
FBQYGCAgICQoKCwsMDAwNDQ0ODg4ODw8ODxAQDxAPDg8OHR4fHxUWFRUVFBQUExMSEREQApYOEA8
REBESEpr+U6EVFBQUExESEAoKCQgICAYGBQUEAwICAQEBAQICAwQFBQYGCAgICQoKCwsMDAwNDQ0
ODg4ODw8ODxAQDxAPDg8OHR4fHxUWFRUVFBQUExMSEREQApYOEA8REBESEpr+U6EVFBQUExESEAo
KCQgICAYGBQUEAwICAQEBAQICAwQFBQYGCAgICQoKCwsMDAwNDQ0ODg4ODw8ODxAQDxAPDg8OHR4
fHxUWFRUVFBQUExMSEREQApYOEA8REBESEpr+U6EVFBQUExESEAoKCQgICAYGBQUEAwICAQEBAQI
CAwQFBQYGCAgICQoKCwsMDAwNDQ0ODg4ODw8ODxAQDxAPDg8OHR4fHxUWFRUVFBQUExMSEREQApY
OEA8REBESEpr+U6EVFBQUExESEAoKCQgICAYGBQUEAwICAQEBAQICAwQFBQYGCAgICQoKCwsMDAw
NDQ0ODg4ODw8ODxAQDxAPDg8OHR4fHxUWFRUVFBQUExMSEREQApYOEA8REBESEpr+U6EVFBQUExE
SEAoKCQgICAYGBQUEAwICAQEBAQICAwQFBQYGCAgICQoKCwsMDAwNDQ0ODg4ODw8ODxAQDxAPDg8
OHR4fHxUWFRUVFBQUExMSEREQApYOEA8REBESEpr+U6EVFBQUExESEAoKCQgICAYGBQUEAwICAQE
BAQICAwQFBQYGCAgICQoKCwsMDAwNDQ0ODg4ODw8ODxAQDxAPDg8OHR4fHxUWFRUVFBQUExMSEQ
ERE
```

```
BAQEBAwMEBQUGBggICAkKCgsLDAwMDQ0NDg4ODg8PDg8PDw8ODw4ODQ4NDA0MCwwLDw0NCwkIBwW
QAaEhiggJCgsMDA0OEBARERITEhQTFBUUFRYVFhUWFRUVFBQUExMSEhEREOsBqxAQERESEhMTFBQ
UFRUVFhUfHx4dFA4ODg8ODw8PDg8PDg4ODg0NDQwMDAsLCgoJCAgIBgYFBQQDAgIBAQEBAgIDBAU
FBgYICAgJCgoQEhETFBQUFaEBrZoSEhEQEQ8QDg8ODQ0LCgoICAcFBQMDAgIDAwUGBggICgoMDA0
OAAUAAAAAA98D9AADAAYADQARABQAAAEXNycxASETETcRNwEjNxczJzkCAqGeJKABHP2HKuAx/nv
7xzRKWwIsliWYAUv+Vf4ZUgGVOQFyMTFWAAAAAAEAAAAAA8gD9AAFAAABETcRASEBqLABcPxwAh/
97aIBcQHVAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAEABwABAAEAAAAAAAIABwAIAAEAAAAAAAM
ABwAPAAEAAAAAAAQABwAWAAEAAAAAAAUACwAdAAEAAAAAAAYABwAoAAEAAAAAAAoALAAvAAEAAAA
AAAsAEgBbAAMAAQQJAAAAAgBtAAMAAQQJAAEADgBvAAMAAQQJAAIADgB9AAMAAQQJAAMADgCLAAM
AAQQJAAQADgCZAAMAAQQJAAUAFgCnAAMAAQQJAAYADgC9AAMAAQQJAAoAWADLAAMAAQQJAAsAJAE
jIHRvb2xiYXJSZWd1bGFydG9vbGJhcnRvb2xiYXJWZXJzaW9uIDEuMHRvb2xiYXJGb250IGdlbmV
yYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAHQ
AbwBvAGwAYgBhAHIAUgBlAGcAdQBsAGEAcgB0AG8AbwBsAGIAYQByAHQAbwBvAGwAYgBhAHIAVgB
lAHIAcwBpAG8AbgAgADEALgAwAHQAbwBvAGwAYgBhAHIARgBvAG4AdAAgAGcAZQBuAGUAcgBhAHQ
AZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB
1AGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMAbwBtAAAAAAAIAAAAAAAAACgA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACwECAQMBBAEFAQYBBwEIAQkBCgELAQwACnNob3ctMDEtd2Y
IaGlkZTEtd2YHem9vbS1pbgh6b29tLW91dAptZWRpYS1wbGF5C21lZGlhLXBhdXNlBHVuZG8ecmV
kbwtmaWx0ZXItbm9uZQZmaWx0ZXIAAA==) format('truetype');
            font-weight: normal;
            font-style: normal;
        }
        .e-toolbar.toggle .e-btn .e-icons {
            font-family: 'toolbar' !important;
            font-size: 16px;
            font-style: normal;
            font-weight: normal;
            font-variant: normal;
            text-transform: none;
        }
        .e-hide-icon::before {
            content: '\a701';
        }
        .e-show-icon::before {
            content: '\a700';
        }
        .e-filter-icon::before {
            content: '\a709';
        }
        .e-filternone-icon::before {
            content: '\a708';
        }
        .e-undo-icon::before {
            content: '\a706';
        }
        .e-redo-icon::before {
            content: '\a707';
        }
        .e-play-icon::before {
            content: '\a704';
        }
        .e-pause-icon::before {
            content: '\a705';
        }
        .e-zoomin-icon::before {
            content: '\a702';
        }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        .e-zoomout-icon::before {
            content: '\a703';
        }
</style>
```

{% previewsample "page.domainurl/code-snippet/toolbar/how-to/toggle-button-cs1" %}

Customize toolbar scroll step in Vue Toolbar component

Toolbar supports to customize the scrolling distance when you click the left and right side navigation icons. we can customize ScrollStep property for scrolling distance. Refer to the following code example.

By using Toolbar scrollStep property, pass a required value to customize toolbar scrollStep.

APP.VUE

```
<template>
  <div id="app">
        <br>
        <ejs-toolbar id="toolbar" width="600" scrollStep="50">
            <e-items>
                <e-item prefixIcon='e-cut-icon tb-icons'
tooltipText='Cut'></e-item>
                <e-item prefixIcon='e-copy-icon tb-icons'
tooltipText='Copy'></e-item>
                <e-item prefixIcon='e-paste-icon tb-icons'
tooltipText='Paste'></e-item>
                <e-item type='Separator'></e-item>
                <e-item prefixIcon='e-bold-icon tb-icons'
tooltipText='Bold'></e-item>
                <e-item prefixIcon='e-underline-icon tb-icons'
tooltipText='Underline'></e-item>
                <e-item prefixIcon='e-italic-icon tb-icons'
tooltipText='Italic'></e-item>
                <e-item prefixIcon='e-color-icon tb-icons'
tooltipText='Color-Picker'></e-item>
                <e-item type='Separator'></e-item>
                <e-item prefixIcon='e-alignleft-icon tb-icons'
tooltipText='Align-Left'></e-item>
                <e-item prefixIcon='e-alignright-icon tb-icons'
tooltipText='Align-Right'></e-item>
                <e-item prefixIcon='e-aligncenter-icon tb-icons'
tooltipText='Align-Center'></e-item>
                <e-item prefixIcon='e-alignjustify-icon tb-icons'
tooltipText='Align-Justify'></e-item>
                <e-item type='Separator'></e-item>
                <e-item prefixIcon='e-bullets-icon tb-icons'
tooltipText='Bullets'></e-item>
                <e-item prefixIcon='e-numbering-icon tb-icons'
tooltipText='Numbering'></e-item>
                <e-item type='Separator'></e-item>
                <e-item prefixIcon='e-ascending-icon tb-icons'
tooltipText='Ascending'></e-item>
                <e-item prefixIcon='e-descending-icon tb-icons'
tooltipText='Descending'></e-item>
                <e-item type='Separator'></e-item>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-item prefixIcon='e-upload-icon tb-icons'
tooltipText='Upload'></e-item>
                <e-item prefixIcon='e-download-icon tb-icons'
tooltipText='Download'></e-item>
                <e-item type='Separator'></e-item>
                <e-item prefixIcon='e-indent-icon tb-icons'
tooltipText='Indent'></e-item>
                <e-item prefixIcon='e-outdent-icon tb-icons'
tooltipText='Outdent'></e-item>
                <e-item type='Separator'></e-item>
                <e-item prefixIcon='e-clear-icon tb-icons'
tooltipText='Clear'></e-item>
                <e-item prefixIcon='e-reload-icon tb-icons'
tooltipText='Reload'></e-item>
                <e-item prefixIcon='e-export-icon tb-icons'
tooltipText='Export'></e-item>
            </e-items>
        </ejs-toolbar>
    <br/>
    </div>
</template>
<script>
import Vue from 'vue';
import { ToolbarPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(ToolbarPlugin);
export default {
  name: 'app',
  data () {
}
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
/* custom code start */
.view-site:before {
  content: "share";
  font-family: "Material Icons";
}
@font-face {
    font-family: 'Material_toolbar';
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tShMAAAEoAAAAVmNtYXDoMOjqAAACDAAAAHhnbHlmIuy
19QAAAswAACNMaGVhZA6okZMAAADQAAAANmhoZWEIUQQkAAAArAAAACRobXR4jAAAAAAAAYAAAAC
MbG9jYYc0kUIAAAKEAAAASG1heHABOwG8AAABCAAAACBuYW1lx/RZbQAAJhgAAAKRcG9zdJZeEVU
AACisAAACGAABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAIwABAAAAAQAAAQsu/F8
PPPUACwQAAAAAANXLJlEAAAAA1csmUQAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAjAbAADgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnIQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQ
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAA
AAAQAAAAEAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAGQAAAAEAAQAAQAA5yH//wA
A5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcACAAJAAoACwAMAA0ADgAPABAAEQASABMAFAAVABY
AFwAYABkAGgAbABwAHQAeAB8AIAAhACIAAAAAADIAjgFwAfgCIAKYAxIDSAO2BRYFMAVcBnIGugb
```

2ByoHQgguCNYJRgn6CiQKiAquCsgMFgzADOYNzg7WDvAQyBEyEaYABwAAAAAD9APzAAMABwAKAA4
AEgAVABkAADchNSElITUhJTkBBSE1ITUhNSEFFxEnITUhDAPo/BgBtgIy/c7+SgG2AjL9zgIy/c7
+Svr6A+j8GAxefV67Pl19Xvr6AfScXgAAAAIAAAAAA/QD9AAEAEgAACUhNxc3AREfDyE/DxEvDyE
PDgOF/PbDisP9gQEBAwQEBgYICAgJCgoLCwsDCgsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQo
KCwsL/PYLCwsKCgkICAgGBgQEAwGz+qf6AYX89gsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQo
KCwsLAwoLCwsKCgkICAgGBgQEAwEBAQEDBAQGBggICAkKCgsLAAACAAAAAAPzA/QAQAC/AAABFQ8
PLw8/Dx8OAQ8ELwErAQ8FFR8FBxcPAxUfBzsBNx8LOwI/Cx8BOwE/Bj0BLwQ/Aic/BC8HKwEHLws
rAg8FArIBAgUGBwkKDAwODxAQERITEhIREQ8PDgwMCgkHBgUCAQECBQYHCQoMDA4PDxEREhITEhE
QEA8ODAwKCQcGBQL+zxUWFhUWfwUFBAUDBANqAgEBAgIDbgMDbwMCAQEBAmkDBAQEBQSEFBYWFxQ
CAgIDBAQEBcwFBAQEAwICAhQXFRYVgAQFBQQEAwNoAgEBAgIDcAEBAQNvAgIBAQEBA2gDBAQFBAW
DFBYWFxIBAgMDAwQFBcwFBAQDBAICAgAJCRIQEBAODgwLCgkHBgQDAQEDBAYHCQoLDA4OEBAQEhI
SEhAQEA4ODAsKCQcGBAMBAQMEBgcJCgsMDg4QEBASAc6ECwwNDjIBAQICA7QEBQQFBAMEUjIyVgM
EBAQFBAWwAwICATMODQwLhAQEBAMCAgECAgIDBAMEhAsMDQ4yAQECAgOwBAQFBAUEAwRSDAwaMlY
DBAQEBQQFsAMCAgEzDg0MC4QEAwQDAgICAgICAwQDAAAAAAMAAAAAA/MD2AAyADUAaQAAJRUfDTs
BPw41LwgPBwMhAScXAQ8GHQEfBQEfBjsBPwYBPwYvBwEDFgIDBAQGBgcICQkKCgsLCwsLCwoKCQg
ICAYGBAQDAQEDBAcMCQoLFCMtFQoJCQcFBHv96gEL04X+4gYFBAQCAgICAgIEBAUBNwcHBwgHCAc
IBwgHCAgHBwYBOAUEAwMCAQEBAQIDAwQFBv4PlwsLCwoKCQkIBwYGBAQDAgIDBAQGBgcICQkKCgs
LCwcPEBAYEBAPHCk3HRAQEBAQEAEIAQrThf7iBgcIBwgHCAgICAgHBwcH/skGBQQEAwIBAQIDBAQ
FBgE3BwcHBwgICAgHCAgHCAcGAfEABQAAAAAD9APzAAMABwALAA8AEwAANyE1ITchNSEnITUhNyE
1ISchNSEMA+j8GN4CLP3U3gPo/BjeAiz91N4D6PwYDF6AW5xefVqAXgAAAAAEAAAAAAP0A/QACQA
TABcAWwAAAQcVMzcXMzUjLwEjFTMbATM1IwElESERBxEfDyE/DxEvDyEPDgFro8ObnnROxOp0nZv
qTij+8AGW/NReAQEDBAQGBggICAkKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJCgo
LCwv89gsLCwoKCQgICAYGBAQDAQENAyOWliO4BSUBK/7VJQFSffzUAywR/PYLCwsKCgkICAgGBgQ
EAwEBAQEDBAQGBggICAkKCgsLCwMKCwsLCgoJCAgIBgYEBAMBAQEBAwQEBgYICAgJCgoLCwAAAAA
CAAAAAAOWA/QAAwBpAAA3ITUhExUfHTsBPx01ESMRDw8vDxEjagMs/NRKAgIDAwUFBgcHCAkJCgs
LCwwNDQ0ODw4PEA8QERARERERERBEQDxAPDg8ODQ0NDAsLCwoJCQgHBwYFBQMDAgKLAQMFBggKCww
ODxARERMTFBQTExEREA8ODAsFCQcGBAKLDH0BsBERERERQEA8QDg8ODg0MDQsLCwoJCQgHBwYFBQQ
DAgEBAgMEBQUGBggICQkKCgsMDAwNDg4ODw8PEBAQERERAb7+RRQTEhIREA8NDQsKCAYFAwEBAwU
GCAoLDQ0PCBASEhMTAcUABQAAAAAD9APXAAIABQANABcAGgAAJTcjASM3ATM3MxczAyMFIQEVITU
hATUhJTMnAgJx4wG/vl/+/Fot+i1a3FD9RgEg/t4Bov7UAST+aAF/6XQobQET//47eHgCM07+XD5
NAaQ/UHMAAAAAAQAAAAAD9ALoAF8AABMhJz8PHxo3Lx8PDycMAbWyDQ0ODg8PDxAQEBERERIREhA
QEBAQDw8PDw4ODg0NDQwMFxYTEhAHBgYGBXUHBwgJCQoLCwwNDQ0PDg8QEBERERITEhMUExQVFBU
VFRgYFxcXFhYVFhQUFBMTEhGwARi6CwsJCggICAYGBgQEAwIBAQEBAgIDBAQFBQYHBwgICQkKFRY
YGhsODg8PDygUFBMTEhISERARDw8PDg0NDAsLCgoICAgGBgQEAwMBAQECAwQFBgcJCQoLDA0ODg+
6AAYAAAAAAA/MD9AA/AGsAqwDrAO8BMwAAARUfDTsBPw09AS8ODw4lHwk7AT8IPQEvByMnByMPBYU
fDz8PLw8PDiUfDz8OPQEvDSsBDw0lESERBxEfDyE/DxEvDyEPDgHhAgMFBQYHCAkKCgsLDA0NDA0
MCwsKCgkIBwYFBQMCAgMFBQYHCAkKCgsLDA0MDQ0MCwsKCgkIBwYFBQMC/scBAQEFBwgKCwYGBwY
GBgwKCAcFAQEBAQUHCAoMBgYGBwYGCwoIBwUBAQHzAQECBAQEBgYGCAcICQkJCgoJCQgJBwgGBgY
EBAMDAQEBAQMDBAQGBgYIBwkICQkKCgkJCQgHCAYGBgQEBAIB/qgBAQMEBAYGBwgICQoKCgsLCws
LCgkJCQcHBwUFAwMCAgMDBQUHBwcJCQkKCwsLCwsKCgoJCAgHBgYEBAMBAlD81F4BAQMDBAUGBgc
HCAkJCQkAyYKCQkJCQgHBwYGBQQEAgEBAQECBAQFBgYHBwgJCQkJCvzaCgkJCQkIBwcGBgUEAwM
BAWQNDAwMCwoKCQgHBgUFAwICAwUFBgcICQoKCwwMDA0NDAwLCwsJCQgHBwUEAwIBAQIDBAUHBwg
JCQsLCwwMMQYGBgsKCQcEAgEBAgQHCQoLBgYGBwYGCwoJBgUCAQECBQYJCgsGBvMJCgkICAgHBwY
FBQQDAwEBAQEDAwQFBQYHBwgICAkKCQoJCQkICAcGBwUFBAMCAQEBAQIDBAUFBwYHCAgJCQkGCws
KCgoJCAgHBgYEBAMBAQEBAwQEBgYHCAgJCgoKCwsLCwsKCQkJBwcHBQUDAwICAwMFBQcHBwkJCQo
LC9/81AMsA/zaCgkJCQkIBwcGBgUEBAIBAQEBAgQEBQUHBwcICQkJCQoDJgoJCQkJCAcHBwUFBAQ
CAQEBAQIEBAUFBwcHCAkJCQkAAAAAAgAAAAADtQP0AAMACgAANyE1IQEjCQEjESFKA2z8lAEG7AG
cAZzs/qAMfQIK/mQBnAFhAAYAAAAAA/QD8wADAAcACwAPABIAFgAANyE1ISUhNSE1ITUhNSE1KQE
RNwMhNSEMA+j8GAG2AjL9zgIy/c4CMv3O/kr6+gPo/BgMXn1efV19Xv4M+gGWXgAFAAAAAAPzA/M
AJQBpAKgArADwAAABFT8bIw8GBR8PNSMvDT8CJw8OHw8TByMPDBc/AzsBHwUzHwYzLxcjJREhEQc
RHw8hPw8RLw8hDw4CKg8QDw8ODg4NDQwMDAwKCwoKCQgJDw0KCQQDAgLxBAUGBggJCgsMDAwKCwo
PDw8PEA8QEBAQEAoKCQkJCQgIBgQEBAUDAQEDBKsKCQgIBwcGBgUDAwMBAQECBAQFBgYICAkJCgo
ICAgGBgQEAwMBAQECBAQFBgYICAkJCgoLCwsYGhsbDw4PDwo
KCQgIBwaUDAsLCgkIBwcGBQQEAgIBAfEBAwMEBQYIBQcGBw8PEA8ODa0NDQ0ODg4ODw8PEA8
PEA8PEA8ODw8ODg0ODQ0MAj8BBAMDBAQFBgcHBwkJCQoLCwsKCQgICAYGBgQEAwIBAQICBAMJEAcHCAgJCh0dHRsaGRc

XCgoKCQgICAcGBgYEBQMDBj781AMsA/zaCgkJCQkIBwcGBgUEAwMBAQEBAgQEBQYGBwcICQkJCQo
DJgoJCQkJCAcHBgYFBAQCAQEBAQIEBAUFBwcHCAkJCQkAAgAAAAAD8wPrAB8AMwAAEw8HHww/BBU
hNSEBNwkBPwcvCDYKCAcGBQMCAQECAwUGBwgKrwgJCgoKCgkINQKQ/Xv+20EBPAGOCgkHBgUDAgE
BAgMFBgcJCtMBoQsMDQ0NDg4ODg0ODQ0NDAuvBgUDAQECBAY0I14BJEH+xAGRCwwMDQ0ODg4ODg4
NDQwMC9QABgAAAAAD9AP0AAMADwATAB0AIQAnAAAlITUhIzMVIxUzFSMVMzUjNyE1ISMzBxUzNSM
3NSM3ITUhJzMVMzUjAQYC7v0S+n0/P328vPoC7v0S+nh4vHh4vPoC7v0S+j4/fWpeID4gPvrbXXw
/P3w/u14gvPoAAAAABQAAAAAD9APbAAIABQANABcAGgAAJTcjAyM3ATM3MxczAyMFIQEVITUhATU
hJzMnAgJx4x6+Xv76Wi39LF3fTwFlAST+3AGk/tIBJP5mw+l0JXMBGP/+N3h4AjRQ/lo+TQGpPk5
zAAABAAAAAAOsA/QACwAAATMDIxUhNSMTMzUhAXGd88gCPJ3zyP3EAx79xNbWAjzWAAAGAAAAAAP
0A9QAAwBDAEcAhwCLAMsAACUhNSEHFR8OPw49AS8ODw4TITUhBxUfDTsBPw09AS8ODw4TITUhBxU
fDj8OPQEvDg8OAQYC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgYGBQQEAwMCAQECAwMEBAUGBgY
HBwgICAgIBwgGBwYFBQUEAwMBAvoC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgYGBQQEAwMCAQE
CAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAvoC7v0S+gIBAwMEBQUFBgcGCAcICAgICAcHBgY
GBQQEAwMCAQECAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAkpeLwgIBwcHBwYFBQUDBAICAQE
BAQICBAMFBQUGBwcHBwgICAgIBwcGBgYFBAQDAwIBAQEBAgMDBAQFBgYGBwcICAFgXS4ICAgHBwY
GBgUEBAMDAgEBAgMDBAQFBgYGBwcICAgICAcHBwcGBQUFAwQCAgEBAQECAgQDBQUFBgcHBwcIAUB
dLggICAcHBgYGBQQEAwMCAQEBAQIDAwQEBQYGBgcHCAgICAgHBwcHBgUFBQMEAgIBAQEBAgIEAwU
FBQYHBwcHCAAAAwAAAAADmQP0AAcAKACNAAABFSE1MxEhESUHFQ8GLwc/Bx8GJysBDw0VERUfDTM
hMz8NNRE1Lw0rAS8OKwEPDQEdAcZb/YQBbAEDBAYHBwkJCQkHBwYEAwEBAwQGBwcJCQkJBwcGBAO
svwkJCQgICAcGBgYEBAMCAgICAwQEBgYGBwgICAkJCQJ8CQkJCAgIBwYGBgQEAwICAgIDBAQGBgY
HCAgICQkJvwMFBQYGBwgICQkJCgoKCwsLCwoKCgkJCQgIBwYGBQUDPoiI/SkC1y4FBQgIBwUEAwE
BAwQFBwgICgkICAcFBQIBAQIFBQcICCQCAgMEBAYGBgcICAgJCQn9KQkJCQgICAcGBgUFBAMCAgI
CAwQFBQYGBwgICAkJCQLXCQkJCAgIBwYGBgQEAwICCgkJCAgIBwYGBQQEAwICAgIDBAQFBgYHCAg
ICQkAAQAAAAAD9ALoAGAAAAExLw8PHxc/Gh8PByERA0QREhMTFBQUFRYWFhcXFxgYFRUVFBUUExQ
TEhMSERERERBAPDg8NDQ0MCwsKCQkIBwd1BQYGBgcQEhMWFwwMDQ0NDg4ODw8PDxAQEBAQEhESERE
QERAQDw8PDg4NDbIBtQIuDw4ODQwLCgkJBwYFBAMCAQEBAwMEBAYGCAgICgoLCwwNDQ4PDw8REBE
SEhITExQUKA8PDw4OGxoYFhUKCQkICAcHBgUFBAQDAgIBAQEBAgMEBAYGBggICAoJCwu6AdAAAAA
OAAAAAAP0A/MAAgAFAAgACwAQABQAFwAbAB4AIQApAC0AMQB1AAABETclFzUXNyMFNyETFQUhEQE
hJRMlMycFMSEnBzcnBxcRBRMDBSUDEy0BEQMlIwUDEQcRHw8hPw8RLw8hDw4CGcj+ZaG3MJb+wM7
+4jQBCv6EAy7+ggEKdP1S3JkBCwEjWemWlvrIATJ0dP7n/up3dwEWAZhy/vQ0/vZyXgEBAwQEBgY
ICAgJCgoLCwsDCgsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAgICQoKCwsL/PYLCwsKCgkICAgGBgQ
EAwEBxv7fWSQ730Bky8v+9QNxAYH+f28BHx2ZmcukmTgJywEeP/7n/ud3dwEZARl3Bv5xAR1ycv7
yAYAR/PYLCwsKCgkICAcHBQUEAwEBAQEDBAUFBwcICAkKCgsLCwMKCwsLCgoJCAgHBwUFBAMBAQE
BAwQEBgYIBwkJCgoLCwAAAAAFAAAAAAP0A/MAAwAHAAsADwATAAA3ITUhJSE1ISUhNSElITUhJSE
1IQwD6PwYAVgCkP1w/qgD6PwYAVgCkP1w/qgD6PwYDF6AW5xefV19XgAAAAAKAAAAAAP0A/MAAwA
HAAsADwATABcAGwAfACMARwAAARUjNSMVIzUjFSM1ARUjNSMVIzUjFSM1JRUjNSMVIzUjFSM1JxE
fByE/BxEvByEPBgOW+j7bP9oDLPo+2z/aAyz6Pts/2l4BAwUGAwgJCgOJCgkJBwYDBAIBAwUGAwg
JCvx3CgkJBwYFAwElvb27u7u7ARrb29vb29v6vLy8vLy8hvyCCwoJBwQGBAIBAwUHBwUJCgOECwo
JBwQGBAIBAwUGCAkKAAAAAAUAAAAAA/QD8wADAAcACwAPABMAADchNSE1ITUhNSE1ITUhNSE1ITU
hDAPo/BgCkP1w/qgD6PwYDF6BV59efV19XgAAAAKAAAAAAP0A00AAwAHAAsADwATABcAGwAfACMAR
/Dx8OExUPBSsBLwU9AT8FOwEfBQMPDy8PPw8fEAE1IwUVHw8zPwMXBy8FDw8fDz8PNS8DNwEzNQE
/BS8PDw4BOAEBAwMEBQYGBwgICQkKCgoKCgoJCQgIBwcGBgUFBAQDAeICAgMDBQUFBQUFAwMCAgI
CAgMDAwUFBQUFBQUDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAgHBwYGBQUEBAMDAeICAgMDBQU
FBQUFAwMCAgICAgMDAwUFBQUFBQUDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAgHBwYGBQUEBAM
DAaftkAV6W/K4BAwUHCAoMDQ4PERETExQUCwsVFBN2dgkKCgoVFhQUExMREQ8ODQwKCAcFAwEBAwUHCAoMDQ4PERE
TExQUFBQTExERDw4NDAoIBwUDAQEEBgd2AV6W/ZYFBAMCAwEBAwUHCAoMDQ4PERETExQUFBQTExE
RDw4NDAoIBwUD1AoKCgkJCAgHBgYFBAMDAQEBAQMDBAUGBgcICAkJCgoKCgoKCQkICAcGBgUFBAM
BAQEBAwMEBQYGBwgICQkKCgEiBQUFAwMCAgICAwMFBQUFBQUDAwICAgIDAwUFASCkCgoJCQgIBwY
GBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAgHBwYGBQUEBAMDAQEBAQMDBAUGBgcICAkJCgq
gqgZAFeMpYKChQTExERDw4NDAoIBwUDAQEEBgd2dgUEAwIDAQEDBQcICgwNDg8RERMTFBQUFBMTEREPDg0
MCggHBQMBAQMFBwgKDA0ODxEREhMUFAsLFRQTdv6iMgJqCQoKChUWFBQTExERDw4NDAoIBwUDAQE
DBQcICgwNDg8RERMTFAADAAAAAAANXA7UAIgBFAJMAAAEzHw4PDisBNRMzHw4PDisBNQMhPxEvDz8
PLxghAkgKCgkJCAgHBwYGBAQEAgEAgQEBAQEDAwQFBgYHBwgJCAkKCeDACgoJCQgIBwcGBgQEBAIBAQE
BAgQEBAYGBwICAkJCgrAwHDDQwMDBcWFRMSEQ8NDAoHBgQBAQIDBAYHBwkKCgsNDA4ODwsLCgo
KCAgIBgYFBQMDAQEBAQECAwQEBAUGDA8QEhQVFgwMDA0ODg8PDxAQEBAQEhESERERERMTExQUFBU
KCQkICAcHBgUFBAQDAgIBAQEBAgMEBAYGBwgICQkKCgoKCgkJCAgHBwYGBQUEBAMDAQEBAQMDBAU
FBQUFAwMCAgICAgMDAwUFBQUFBQUDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAgHBwYGBQUEBAM
DAaftkAV6W/K4BAwUHCAoMDQ4PERETExQUCwsVFBN2dgkKCgoVFhQUExMREQ8ODQwKCAcFAwEBAwUHCAoMDQ4PERE
TExQUFBQTExERDw4NDAoIBwUDAQEEBgd2AV6W/ZYFBAMCAwEBAwUHCAoMDQ4PERETExQUFBQTExE
RDw4NDAoIBwUD1AoKCgkJCAgHBgYFBAMDAQEBAQMDBAUGBgcICAkJCgoKCgoKCQkICAcGBgUFBAM
BAQEBAwMEBQYGBwgICQkKCgEiBQUFAwMCAgICAwMFBQUFBQUDAwICAgIDAwUFASCkCgoJCQgIBwY
GBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAgHBwYGBQUEBAMDAQEBAQMDBAUGBgcICAkJCgq
gqgZAFeMpYKChQTExERDw4NDAoIBwUDAQEEBgd2dgUEAwIDAQEDBQcICgwNDg8RERMTFBQUFBMTEREPDg0
MCggHBQMBAQMFBwgKDA0OPERETExQUFBMTEREPDg0MCggHBQMBAQMFBwgKDA0O

MDg8REhQUFhcYGBERERAQEA4ODgwMDAoJCQcICQkKCgoLDAsMDAwMDQwNDQwNDQwMCwwLCxQUERA
ODQoFAwQDAgEBAQAABQAAAAAD9APzAAMABwALAA8AEwAANyE1ITUhNSE1ITUhNSE1ITUhNSEMA+j
8GAPo/BgD6PwYA+j8GAPo/BgMXn1enF59XX1eAAAAAAEAAAAAA9QD1ADUAAATHx8/DxcRIRcPDy8
fPx8fDzMvHw8eKwECAwQFBggICQoMDA0ODhAQERISExQUFRUWFhcXGBgYGBgXFxcWFhUVFBQTEhI
REIr+ZrsMDA0ODg4PEBAQEBESERISEhIREhEQEQ8QDw8ODg0NDAwLCgoJCQgHBgYEBAQCAQEBAQI
EBAQGBgcICQkKCgsMDA0NDg4PDxAPERAREhESEhwcGxoaGBgWFRQSEQ8OCwp7BQYHCAgJCQoLCww
NDQ4ODg8QEBERERISEhMTFBMUFRQYGBgXFxYWFRUUFBMSEhEQEA4ODQwMCgkICAYFBAMCAgAYGBc
XFxYWFRUUFBMSEhEQEA4ODQ0LCgoICAYFBAMCAQECAwQFBggICgoLDQ0ODhCKAZq7DAsLCgkJCAc
HBQUEAwMBAQEBAgQEBAYGBwgICgkLCwwMDQ0ODg8PDxAREBESERISEhIREhEQERAPDw8ODg0NDAw
LCwkKCAgHBgYEBAQCAQECAwUICQsNDxASExUWFxgaExITERIREBAQDw8ODg0NDAsLCgoJCAcHBgY
EBAMCAQEBAgMEBQYICAoKCw0NDg4QEBESEhMUFBUVFhYXFxcYAAAAAgAAAAD8gP0AGcA7gAAARU
PGC8YPQE/FzsBHxcFHx8/DxcVATcBIyc/Dj0BLx0rAQ8dAoABAgIDAwQFBQUNDxATExYLCwwMDAw
NDQ0NDQ0NDQwNDAsMCxUUEhAPDQUFBQQDAwMBAQEBAwMDBAUFBQ0PEBIUFQsMCwwNDA0NDQ0NDQ0
NDAwMDAsLFhMTEA8NBQUFBAMDAgIB/Y0BAQMDBAUGBggICQkLCwsNDA4ODg8QEBARERISEhMTExE
REBEQEBAQDw8ODg4ODA0OAR1W/uMuDgoKCQkIBwYGBgQEAwMCAQICAwQFBgcHCAkKCgsMDA0NDg8
PDxARERERER

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAjAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4BDwEQAREBEgE
TARQBFQEWARcBGAEZARoBGwEcAR0BHgEfASABIQEiASMBJAAQVGV4dF9PdXRkZW50XzAwMQtQaWN
0dXJlXzAwMQxTZXR0aW5nc18wMDEQQ29sb3JfcGlja2VyXzAwMhBBbGlnbl9DZW50ZXJfMDA2CEx
pbmVfMDAxDVVuZGVybGluZV8wMDEMU29ydF9aLUFfMDAxCFVuZG9fMDAxEENoYXJ0X2J1YmJsZV8
wMDELRG93bmxvYWRfMDAPVGV4dF9pbmRlbnRfMDAxEkNoYXJ0X0RvdWdobnV0XzAwMQlDbGVhcl8
wMDINTnVtYmVyaW5nXzAwMQxTb3J0X0EtWl8wMDEKSXRhbGljXzAwMQtCdWxsZXRzXzAwMQlQYXN
0ZV8wMDEIUmVkb18wMDEPQ2hhcnRfcmFkYXJfMDAxD0FsaWduX1JpZ2h0XzAwMQlUYWJsZV8wMDE
OQWxpZ25fTGVmdF8wMDEITWVudV8wMDEHQ3V0XzAwMghCb2xkXzAwMRFBbGlnbl9KdXN0aWZ5XzA
wMQpSZWxvYWRfMDAxClNlYXJjaF8wMDEKVXBsb2FkXzAwMQpEZXNpZ25fMDA1CkV4cG9ydF8wMDE
IQ29weV8wMDIAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
@font-face {
        font-family: 'Material_toolbar2';
        src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMjhtRPcAAAEoAAAAVmNtYXDiGOJwAAABnAAAAEBnbHlmSXI
howAAAewAAAacaGVhZBR10/MAAADQAAAANmhoZWEIRwQIAAAArAAAACRobXR4HAAAAAAAAYAAAAA
cbG9jYQUwBn4AAAHcAAAAEG1heHABGQC2AAABCAAAACBuYW1lBOSmewAACIgAAAKdcG9zdMbb77w
AAAsoAAAAaQABAAAEAAAAAFwEAAAAAAD6gABAAAAAAAAAAAAAAAAAAAAABwABAAAAAQAAyBsHLV8
PPPUACwQAAAAANi2x4EAAAAA2LbHgQAAAAAD6gP0AAAACAACAAAAAAAAAAEAAAAHAKoACAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4gDiBQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAAAAAIAAAADAAAAFAADAAEAAAAUAAQALAAAAAQABAABAADiBf/
/AADiAP//AAAAAQAEAAAAAAQACAAMABAAFAAYAAAAAAMABKAGmAggCygNOAAIAAAAAA3cD9ABUAKk
AAAEfBw8eIzUHFzU/Hy8HJQ8fHwc3Lwc/HjMVNycC+AcHBgQEAwEBAQECAgMEBAUFBwYHCAkJCgo
KCwsLDAwMDQ0NDQ4ODg+6uhQTExMSERIQERAPDw8ODg0NDAsLCQkIBwcFBQQEAgIBAQQEBwgKDA3
+xBQTExMSERIQERAPDw8ODg0NDAsLCQkIBwcGBAUDAgIBAQMFBwgKDA1ECAYGBAQDAQEBAQICAwQ
EBQUGBwcICQkKCgoLCwsMDAwNDQ0NDg4OD7q6AoAPEA8QEBAREQ4ODg0NDQwNCwwLCwsLCgoJCQg
IBwYGBQUEBAMCAgGItbeIAQICAwQFBgYHCAgKCgsLDQ0NDg4PDw8QEBEREhISExMbGhkZFxgWFqo
BAgIDBAUGBgcICAoKCgwMDQ4ODg8PDxAQERESEhITExsaGRkXGBYWQg8PDxAQERERDg4ODQ0NDQw
MCwwLCgsKCgkJCAgHBgYFBQQEAwICAYi1twAAAAMAAAAA8ED9AADAAcAUwAAJTM1IyURIRE3KwE
PDhEfDjMhMz8OES8OKwE1IxUhNSMCAPv7AV39RjExCwoKCQkICAgGBgUEAwIBAQEBAgQEBQYHBwg
ICQkKCgoCugoKCgkJCAgHBwYFBAQCAQEBAQIEBAUGBwcICAkJCgoKMWX+cmXT+8f93AIk+gEDAwQ
FBQcHCAgJCQoLCv1GCwoJCQkICAgHBgUEAwICAgIDBAUGBwgICAkJCQoLAroKCwoJCQgIBwcFBQQ
DAwFlZWUAAAAHAAAAAAPBA/QAAwAHAAsADwATABcAYwAAATM1IwczNSMHMzUjBzM1IwczNSMlESE
RNysBDw4RHw4zITM/DhEvDisBNSMVITUjAuRWVolVVYpWVolVVYlVVQKe/UYxMQsKCgkJCAgIBgY
FBAMCAQEBAQIEBAUGBwcICAkJCgoKAroKCgoJCQgIBwcGBQQEAgEBAQECBAQFBgcHCAgJCQoKCjF
l/nJlAWZVVVVVVVVVVVXa/dwCJPoBAwMEBQUHBwgICQkKCwr9RgsKCQkJCAgIBwYFBAMCAgICAwQ
FBgcICAgJCQkKCwK6CgsKCQkICAcHBQUEAwMBZWVlAAACAAAAAAPBA/QAAwBPAAABESERNysBDw4
RHw4zITM/DhEvDisBNSMVITUjA139RjExCwoKCQkICAgGBgUEAwIBAQEBAgQEBQYHBwgICQkKCgo
CugoKCgkJCAgHBwYFBAQCAQEBAQIEBAUGBwcICAkJCgoKMWX+cmUClf3cAiT6AQMDBAQGBwcICAk
JCgsK/UYLCgkJCQgICAcGBQQDAgICAgMEBQYHCAgICQkJCgsCugoLCgkJCAgHBwYEBAMDAWVlZQA
ABwAAAAAD6gPhAAMABwAnAEcASwBPAJoAACUhFSE1IRUhJQ8HHwczPwYvBiUPBh0BHwYzPwU9AS8
FNxEhEQERIRElRU7AR8NEQ8PIxUhNSMvDxE/DjsBAVoBT/6xAU/+sQItBgcFBQQCAgEBAgIEBQU
HBgcGBgUDAwIBAQIDAwUGBvz4BwYFBQQDAgIDBAUFBgcHBgUFBAMCAgMEBQUGbwIL/goB3v31Aj5
rCgkKCAkIBwcGBQUEAwIBAQEDAwUFBQcHCAgICgkKClX9nFUKCgkKCAgIBwcFBQUDAwEBAQEDAwU
FBQcHCAgICgkKCmvIKZwoVgEBAgMEBAYGBQUFBAMDAQEBAgMEBQUHBQUEBAQCAgEBAQIDBAQGBgU
FBQQDAwEBAQIDBAUFBwUFBAQEAgIe/swBNAIw/oEBfyvJAgIDBQQGBgcICAkJCgr+fwoKCQkJCAg
HBwUFBQMDAQG7uwEBAwMFBQUHBwgICQkJCgoBdwoKCQkJCAgHBgYFBAQCAgAAAAgAAAAAA8ED8wA
DAAcACwAPABMAFwAbAGcAAAEzNSMHMzUjBzM1IyUzNSMHMzUjBzM1IyURIRE3KwEPDRUDHw4zITM
/DhEvDisBNSMVITUjApZkZMhkZMdjYwGPZGTIZGTHY2MCV/1FMjMKCgkKCAgIBwcFBQQEAgIBAQE
DAwQFBgcHBwkICgkKCgK8CgoJCQkICAcHBQUFAwMBAQEBAwMFBQUHBwgICQkJCgoyZP5wYwEGZGR
kZGRkZGRkZGRk/dsCJfkCAgMFBQUHBwgICQkJCgr9RQsKCQkJCAgHBgYFBAQCAgICBAQFBgYHCAg
JCQkKCwK7CgoJCQkICAcHBQUFAwICZGRkAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAEAEQABAAE
AAAAAAAAIABwASAAEAAAAAAAAMAEQAZAAEAAAAAAAAQAEQAqAAEAAAAAAAAUACwA7AAEAAAAAAAAYAEQB
GAAEAAAAAAAoALABXAAEAAAAAAAsAEgCDAAMAAQQJAAAAAgCVAAMAAQQJAAEAIgCXAAMAAQQJAAI
```

```
ADgC5AAMAAQQJAAMAIgDHAAMAAQQJAAQAIgDpAAMAAQQJAAUAFgELAAMAAQQJAAYAIgEhAAMAAQQ
JAAoAWAFDAAMAAQQJAAsAJAGbIE1hdGVyaWFsX3Rvb2xiYXIyUmVndWxhck1hdGVyaWFsX3Rvb2x
iYXIyTWF0ZXJpYWxfdG9vbGJhcjJWZXJzaW9uIDEuME1hdGVyaWFsX3Rvb2xiYXIyRm9udCBnZW5
lcmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIAB
NAGEAdABlAHIAaQBhAGwAXwB0AG8AbwBsAGIAYQByADIAUgBlAGcAdQBsAGEAcgBNAGEAdABlAHI
AaQBhAGwAXwB0AG8AbwBsAGIAYQByADIATQBhAHQAZQByAGkAYQBsAF8AdABvAG8AbABiAGEAcgA
yAFYAZQByAHMAaQBvAG4AIAAxAC4AMABNAGEAdABlAHIAaQBhAGwAXwB0AG8AbwBsAGIAYQByADI
ARgBvAG4AdAAgAGcAZQBuAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwB
pAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8
AbgAuAGMAbwBtAAAAAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABwECAQMBBAEFAQY
BBwEIAAhNVF9Bc3luYwhNVF9Ub2RheQdNVF9XZWVrB01UX1llYXIKcHJpbnRlci0wMQhNVF9tb25
0aAAAAAAA) format('truetype');
        font-weight: normal;
        font-style: normal;
}
.e-tbar-btn .tb-icons {
  font-family: "Material_toolbar";
  speak: none;
  font-size: 16px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
}
#toolbar_Popalways .e-tbar-btn .e-icons {
  font-family: "Material_toolbar2", "e-icons";
}
.e-bigger .e-tbar-btn .tb-icons {
  font-size: 18px;
}
.e-tbar-menu-icon:before {
  content: "\e718";
}
.e-tbar-search-icon:before {
  content: "\e71d";
}
.e-tbar-settings-icon:before {
  content: "\e702";
}
.e-tbar-section .e-toolbar .e-toolbar-item.e-template {
  cursor: default;
}
.e-def-avator {
  margin: 8px;
  border-radius: 50%;
  width: 40px;
  font-size: 20px;
  border: 2px solid #666;
}
.e-def-avator span {
  text-align: center;
  line-height: 40px;
  display: block;
}
.e-mail-header,
.e-mail-subject,
.e-mail-description,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.e-mail-id {
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
}
.e-folder {
  text-align: center;
}
.e-mail-header,
.e-mail-subject,
.e-folder {
  font-weight: 500;
  font-style: normal;
}
.e-mail-item {
  margin-top: 8px;
  border-bottom: 1px solid #d3d3d3;
  height: 100%;
  overflow: hidden;
}
.e-mail-image {
  margin-right: 10px;
  float: left;
  height: 60px;
}
.e-cut-icon:before {
  content: "\e719";
}
.e-copy-icon:before {
  content: "\e721";
}
.e-paste-icon:before {
  content: "\e712";
}
.e-color-icon:before {
  content: "\e703";
}
.e-bold-icon:before {
  content: "\e71a";
}
.e-underline-icon:before {
  content: "\e706";
}
.e-alignleft-icon:before {
  content: "\e717";
}
.e-alignright-icon:before {
  content: "\e715";
}
.e-aligncenter-icon:before {
  content: "\e704";
}
.e-alignjustify-icon:before {
  content: "\e71b";
}
.e-upload-icon:before {
  content: "\e71e";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
.e-download-icon:before {
  content: "\e70a";
}
.e-indent-icon:before {
  content: "\e70b";
}
.e-outdent-icon:before {
  content: "\e700";
}
.e-clear-icon:before {
  content: "\e70d";
}
.e-reload-icon:before {
  content: "\e71c";
}
.e-export-icon:before {
  content: "\e720";
}
.e-italic-icon:before {
  content: "\e710";
}
.e-bullets-icon:before {
  content: "\e711";
}
.e-numbering-icon:before {
  content: "\e70e";
}
.e-ascending-icon:before {
  content: "\e70f";
}
.e-descending-icon:before {
  content: "\e707";
}
.e-design-icon:before {
  content: "\e71f";
}
.e-table-icon:before {
  content: "\e716";
}
.e-picture-icon:before {
  content: "\e701";
}
.e-redo-icon:before {
  content: "\e713";
}
.e-doughnut-icon:before {
  content: "\e70c";
}
.e-bubble-icon:before {
  content: "\e709";
}
.e-line-icon:before {
  content: "\e705";
}
.e-radar-icon:before {
  content: "\e714";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
.e-undo-icon:before {
  content: "\e708";
}
.e-tool-name {
  font-style: normal;
  font-size: 16px;
  cursor: default;
}
.e-icon-week:before {
  content: "\e202";
}
.e-print:before {
  content: "\e204";
}
.e-icon-month:before {
  content: "\e205";
}
.e-icon-year:before {
  content: "\e203";
}
.e-reccurence-icon:before {
  content: "\e200";
  font-weight: 700;
}
.e-icon-day:before {
  content: "\e201";
}
@media only screen and (min-width: 1224px) {
  .e-tbar-section .e-sample-resize-container {
    width: 700px;
  }
}
@media only screen and (min-width: 1824px) {
  .e-tbar-section .e-sample-resize-container {
    width: 50%;
  }
}
.e-tbar-section .e-sample-resize-container {
  margin: auto;
  height: 40px;
  max-width: 700px;
}
.material .e-bigger .e-toolbar .e-tbar-btn .e-icons {
  font-size: 18px;
}
.material .e-toolbar .e-tbar-btn .e-icons {
  font-size: 16px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/toolbar/how-to/scrollstep-cs1" %}

Add link to toolbar item in Vue Toolbar component

You can add link inside Toolbar using Vue **template**. Follow the below guidelines for add link as template in Toolbar.

- Declare a template in the **template** section of the ".vue" file. An empty object "data" needs to be initialized in the data option of the default export object in **script** section.
- The template function needs to be assigned to the content property of the EJ2 Vue Toolbar Component.

APP.VUE

```
<template>
    <div id="app">
    <ejs-toolbar >
           <e-items>
             <e-item text='Cut'></e-item>
             <e-item text='Copy' ></e-item>
             <e-item type='Separator'></e-item>
             <e-item text='Paste' ></e-item>
             <e-item type='Separator'></e-item>
             <e-item :template='AnchorTemplate'></e-item>
          </e-items>
      </ejs-toolbar>
  </div>
</template>
<script>
import Vue from 'vue';
import { ToolbarPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(ToolbarPlugin);
var AnchorData = Vue.component("demo", {
  template: '<a target="_blank"
href="https://ej2.syncfusion.com/vue/documentation/toolbar/getting-
started/">Anchor Toolbar Link</a>',
  data() {
    return {
      data: {}
    };
  }
});
export default {
  name: 'app',
  data: function(){
    return {
      AnchorTemplate: function () {
        return {
          template: AnchorData
        }
      }
   }
 }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/toolbar/how-to/toolbar-link-cs1" %}

Render other components in toolbar using template in Vue Toolbar component

You can render other components inside Toolbar using Vue **template**. Through this, we can add content as other components directly with all functionalities to our Toolbar. Follow the below guidelines for using the other components as template in Toolbar.

- Declare a template in the **template** section of the ".vue" file. An empty object "data" needs to be initialized in the data option of the default export object in **script** section.
- The template function needs to be assigned to the content property of the EJ2 Vue Toolbar Component.

APP.VUE

```
<template>
    <div id="app">
    <ejs-toolbar >
           <e-items>
             <e-item :template='Template1' ></e-item>
             <e-item text='Italic' ></e-item>
             <e-item text='Underline' ></e-item>
             <e-item type='Separator'></e-item>
             <e-item :template='Template2'></e-item>
          </e-items>
      </ejs-toolbar>
  </div>
</template>
<script>
import Vue from 'vue';
import { ToolbarPlugin } from '@syncfusion/ej2-vue-navigations';
Vue.use(ToolbarPlugin);
import { DatePickerComponent,CalendarComponent,
DatePickerPlugin,CalendarPlugin } from '@syncfusion/ej2-vue-calendars';
Vue.use(DatePickerPlugin);
Vue.use(CalendarPlugin);
import { NumericTextBoxComponent,NumericTextBoxPlugin } from
"@syncfusion/ej2-vue-inputs";
Vue.use(NumericTextBoxPlugin);
export default {
  name: 'app',
  data: function(){
    return {
      Template1: function () {
        return {
          template: Vue.component('NumericTextBoxComponent', {
            template: '<ejs-numerictextbox value="10"></ejs-
numerictextbox>',
            data() { return { }; }
          })
        }
      },
        Template2: function () {
        return {
          template: Vue.component('DatePickerComponent', {
```

```
            template: ' <ejs-datepicker :min="minDate" :max="maxDate"
:value="dateVal" ></ejs-datepicker>',
            data() { return { minDate : new Date("05/09/2017"),
                              maxDate : new Date("05/15/2017"),
                              dateVal : new Date("05/11/2017") }; }
          })
        }
      },
    }
  }
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-buttons/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-
calendars/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/toolbar/how-to/direct-components-cs1" %}

Tooltip

Getting Started with the Vue Tooltip Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Tooltip component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

vue create quickstart

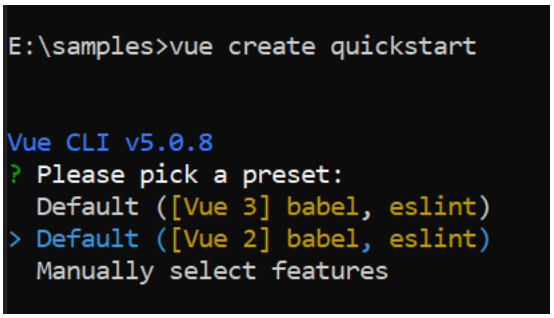
Copyright © 2001 -2024 Syncfusion Inc.

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Tooltip component as an example. Install the @syncfusion/ej2-vue-popups package by running the following command:

`bash

npm install @syncfusion/ej2-vue-popups --save

`

or

`bash

yarn add @syncfusion/ej2-vue-popups

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Tooltip component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

We can also use CRG to generate combined component styles.

Add Syncfusion Vue component

Follow the below steps to add the Vue Tooltip component using Composition API or Options API:

1\. First, import and register the Tooltip component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { TooltipComponent as EjsTooltip } from "@syncfusion/ej2-vue-popups";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { TooltipComponent } from "@syncfusion/ej2-vue-popups";
export default {
components: {
'ejs-tooltip': TooltipComponent
}
}
</script>
```

2\. In the template section, define the Tooltip component with the content property

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-tooltip ref="tooltip" content='Tooltip content' >
<span>Show Tooltip</span>
</ejs-tooltip>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div id='container'>
      <ejs-tooltip class="tooltipcontainer" content='Tooltip content'
target="#target">
        <div style="display: inline-block; position: relative; left:
50%;top: 100px;transform: translateX(-50%);">
          <ejs-button id='target'>Show Tooltip</ejs-button>
        </div>
      </ejs-tooltip>
    </div>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script setup>
import { TooltipComponent as EjsTooltip } from "@syncfusion/ej2-vue-popups";
import { ButtonComponent as EjsButton } from "@syncfusion/ej2-vue-buttons";
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div id='container'>
      <ejs-tooltip class="tooltipcontainer" content='Tooltip content'
target="#target">
        <div style="display: inline-block; position: relative; left:
50%;top: 100px;transform: translateX(-50%);">
          <ejs-button id='target'>Show Tooltip</ejs-button>
        </div>
      </ejs-tooltip>
    </div>
  </div>
</template>
<script>
import { TooltipComponent } from "@syncfusion/ej2-vue-popups";
import { ButtonComponent } from "@syncfusion/ej2-vue-buttons";
export default {
  components: {
    'ejs-tooltip': TooltipComponent,
    'ejs-button': ButtonComponent
  }
}
</script>
<style>
  @import "../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/tooltip/getting-started-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

See Also

Positioning Tooltip

Tooltip Open Mode

Customize the Tooltip

Getting Started with the Vue Tooltip Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Tooltip component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Copyright © 2001 -2024 Syncfusion Inc.

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Tooltip component as an example. To use the Vue Tooltip component in the project, the @syncfusion/ej2-vue-popups package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-popups --save

`

or

`bash

yarn add @syncfusion/ej2-vue-popups

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Tooltip component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Tooltip component using Composition API or Options API:

1.First, import and register the Tooltip component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { TooltipComponent as EjsTooltip  } from "@syncfusion/ej2-vue-
popups";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { TooltipComponent  } from "@syncfusion/ej2-vue-popups";
//Component registration
export default {
name: "App",
components: {
"ejs-tooltip": TooltipComponent
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

2.Add the component definition in template section.

~/SRC/APP.VUE

```
<template>
<ejs-tooltip content='Tooltip content' >
<span>Show Tooltip</span>
</ejs-tooltip>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-tooltip content='Tooltip content' target="#target">
<div style="display: inline-block; position: relative; left: 50%;top:
100px;transform: translateX(-50%);">
<button id='target' class="e-control e-btn e-lib">Show Tooltip</button>
</div>
</ejs-tooltip>
</div>
</template>
<script setup>
import { TooltipComponent as EjsTooltip  } from "@syncfusion/ej2-vue-
popups";
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-tooltip content='Tooltip content' target="#target">
<div style="display: inline-block; position: relative; left: 50%;top:
100px;transform: translateX(-50%);">
<button id='target' class="e-control e-btn e-lib">Show Tooltip</button>
</div>
</ejs-tooltip>
</div>
</template>
<script>
import { TooltipComponent  } from "@syncfusion/ej2-vue-popups";
//Component registration
export default {
name: "App",
components: {
"ejs-tooltip": TooltipComponent
},
data() {
return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
},
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

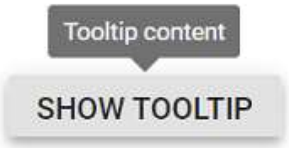
or

`bash

yarn run dev

`

The output will appear as follows:



Tooltip Positioning

Tooltips can be attached to 12 static locations around the target.

On initializing the Tooltip, you can set the position property with any one of the following values:

- TopLeft
- TopCenter
- TopRight
- BottomLeft
- BottomCenter
- BottomRight
- LeftTop
- LeftCenter
- LeftBottom
- RightTop

Copyright © 2001 -2024 Syncfusion Inc.

- RightCenter
- RightBottom

By default, Tooltip is placed at the TopCenter of the target element.

In the following example, the position property set as BottomCenter.

COMPOSITION API (~/SRC/APP.VUE)

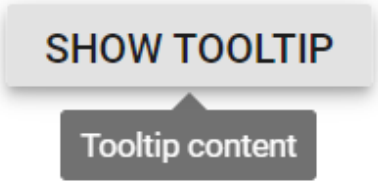
```
<template>
<div id="app">
<ejs-tooltip content="Tooltip content" :position='position'
target="#tooltip">
<div id='container'>
<ejs-button id="tooltip">Show tooltip</ejs-button>
</div>
</ejs-tooltip>
</div>
</template>
<script setup>
import { TooltipComponent as EjsTooltip  } from "@syncfusion/ej2-vue-
popups";
import { ButtonComponent as EjsButton } from '@syncfusion/ej2-vue-buttons';
const position = 'BottomCenter';
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#container {
display: inline-block;
position: relative;
left: 50%;
margin-top: 100px;
transform: translateX(-50%);
height: 200px;
}
#tooltip {
display: inline-block;
margin: 30px 130px 30px 50px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<ejs-tooltip content="Tooltip content" :position='position'
target="#tooltip">
<div id='container'>
<ejs-button id="tooltip">Show tooltip</ejs-button>
</div>
</ejs-tooltip>
</div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { TooltipComponent  } from "@syncfusion/ej2-vue-popups";
import { ButtonComponent } from '@syncfusion/ej2-vue-buttons';
//Component registration
export default {
name: "App",
components: {
"ejs-tooltip": TooltipComponent,
"ejs-button": ButtonComponent
},
data() {
return {
position: "BottomCenter"
};
},
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#container {
display: inline-block;
position: relative;
left: 50%;
margin-top: 100px;
transform: translateX(-50%);
height: 200px;
}
#tooltip {
display: inline-block;
margin: 30px 130px 30px 50px;
}
</style>
```



For migrating from Vue 2 to Vue 3, refer to the migration documentation.

Copyright © 2001 -2024 Syncfusion Inc.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Setting dimension in Vue Tooltip component

Height and width

The Tooltip can either be assigned auto height and width values or specific pixel values. The `width` and `height` properties are used to

set the outer dimension of the Tooltip element. The default value for both the properties is `auto`. It also accepts string and number values in pixels.

The following sample explains how to set dimensions for the Tooltip.

APP.VUE

```
<template>
   <div id='app'>
      <div id='container'>
        <ejs-tooltip id="tooltip" content="This tooltip has 180px width and
40px height" width='180px' height='40px' target="#target">
          <div style="display: inline-block; position: relative; left:
50%;top: 100px;transform: translateX(-50%);">
              <ejs-button id='target'>Show Tooltip</ejs-button>
          </div>
        </ejs-tooltip>
      </div>
    </div>
</template>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(TooltipPlugin);
Vue.use(ButtonPlugin);
export default {
    data() {
    return {
    };
    }
}
</script>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/tooltip/dimensions/height-width-cs1" %}

Scroll mode

When `height` is specified with a certain pixel value and the Tooltip content overflows, the scrolling mode gets enabled.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <div id='app'>
    <ejs-tooltip id='tooltip' :content='content' target='#target'
width='300px' height='60px' isSticky='true'>
      <p>A green home is a type of house designed to be  <a
id="target"><u>environmentally friendly</u></a>
        and sustainable. And also focuses on the efficient use of "energy,
water, and building materials." As green homes
        have become more prevalent we have also seen the emergence of green
affordable housing.
      </p>
    </ejs-tooltip>
  </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  #tooltipContent {
    padding: 0 10px;
    font-weight: 400;
    font-size: 12px;
    line-height: 22px;
  }
</style>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
Vue.use(TooltipPlugin);
var demoVue = Vue.component("demo", {
  template: `
    <div id="tooltipContent"><p><strong>Environmentally friendly</strong> or
<strong>environment-friendly</strong>, <i>(also referred to as eco-friendly,
nature-friendly, and green)</i> are marketing and sustainability terms
referring to goods and services, laws, guidelines and policies that inflict
reduced, minimal, or no harm upon ecosystems or the environment.</p></div>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data: function() {
    return {
      content: function () {
        return { template : demoVue}
      }
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/dimensions/scroll-mode-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

The scrolling mode can best be seen when the sticky mode of the Tooltip is enabled. To enable sticky mode, set the isSticky property to true.

Content in Vue Tooltip component

A text or a piece of information assigned to the Tooltip's content property will be displayed as the main text stream of the Tooltip. It can be a string or a template content. If the content property is not provided with any specific value, then it takes the value assigned to the title attribute of the target element on which the Tooltip was initialized. The content can also dynamically be assigned to the Tooltip via Fetch.

Template content

Any text or image can be added to the Tooltip, by default. To customize the Tooltip layout or to create your own visualized element on the

Tooltip, template can be used.

Refer to the following code example to add formatted HTML content to the Tooltip.

APP.VUE

```
<template>
   <div id='app'>
    <ejs-tooltip id='tooltip' :content='content' target='#target'>
        <p>A green home is a type of house designed to be  <a
id="target"><u>environmentally friendly</u></a>
          and sustainable. And also focuses on the efficient use of "energy,
water, and building materials." As green homes
          have become more prevalent we have also seen the emergence of
green affordable housing.
        </p>
      </ejs-tooltip>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  #tooltipContent {
    padding: 10px;
    font-weight: 400;
    font-size: 14px;
    line-height: 22px;
  }
</style>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
Vue.use(TooltipPlugin);
var demoVue = Vue.component("demo", {
  template: `
    <div id="tooltipContent"><p><strong>Environmentally friendly</strong> or
<strong>environment-friendly</strong>, <i>(also referred to as eco-friendly,
nature-friendly, and green)</i> are marketing and sustainability terms
referring to goods and services, laws, guidelines and policies that inflict
reduced, minimal, or no harm upon ecosystems or the environment.</p></div>`,
  data() {
    return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      data: {}
    };
  }
});
export default {
  data: function() {
    return {
      content: function () {
        return { template : demoVue}
      }
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/content-cs1" %}

Dynamic content via Fetch

The Tooltip content can be dynamically loaded by making use of the Fetch call. The Fetch request is usually made within the `beforeRender` event of the Tooltip, and then the Tooltip's `content` is assigned the value retrieved on it's success.

Note: The Tooltip **target** property includes a unique identifier used to associate Tooltips with specific elements on a webpage or application interface. When setting the Tooltip **target** value as a GUID (Globally Unique Identifier), it's important to note that the GUID must start with a combination of **letters** before the numeric portion of the GUID. For example, **target: '#' + ' tooltip'+ '96ad88bd-294c-47c3-999b-a9daa3285a05'**.

APP.VUE

```
<template>
  <div id='app'>
    <ejs-tooltip style="display:block;" class="e-prevent-select"
:content='content' target='.target' position='RightCenter'
:beforeRender='onBeforeRender'>
      <div id='container'>
        <h4>National Sports</h4>
        <div id="targetContainer" class="e-prevent-select">
            <div id="countrylist">
                <ul>
                    <li class="target" title="1"><span>Australia</span></li>
                    <li class="target" title="2"><span>Bhutan</span></li>
                    <li class="target" title="3"><span>China</span></li>
                    <li class="target" title="4"><span>Cuba</span></li>
                    <li class="target" title="5"><span>India</span></li>
                    <li class="target"
title="6"><span>Switzerland</span></li>
                    <li class="target" title="7"><span>United
States</span></li>
                </ul>
            </div>
        </div>
      </div>
    </ejs-tooltip>
  </div>
```

```
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#container {
  width: 350px;
  position: relative;
  left: 50%;
  transform: translateX(-25%);
}
#countrylist {
  padding: 5px;
}
#countrylist ul {
  border: 1px solid #c4c4c4;
  box-sizing: border-box;
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 100px;
}
#countrylist li {
  padding: 10px;
}
#countrylist li:hover {
  background-color: #ececec;
}
.contentWrap {
  line-height: 16px;
  padding: 3px 0;
}
</style>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
import { Fetch } from '@syncfusion/ej2-base';
Vue.use(TooltipPlugin);
export default {
  data: function() {
    return {
      content : "Loading..",
    };
  },
  methods: {
       onBeforeRender: function(args){
  let fetchApi: Fetch = new Fetch('./tooltipdata.json', 'GET');
    fetchApi.send().then((result)  =>  {
            for (var i = 0; i < result.length; i++) {
                 if (result[i].Id === args.target.getAttribute('data-
content')) {
                    /* tslint:disable */
                    this.content = "<div class='contentWrap'><div
class='def'>" + result[i].Sports + "</div></div>";
                    /* tslint:enable */
                }
                //  this.dataBind();
            }
```

```
        }, (reason) => {
            this.content = reason;
            //  this.dataBind();
        });
       }
   }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/ajax-content-cs1" %}

Position in Vue Tooltip component

Tooltips can be attached to 12 static locations around the target. On initializing the Tooltip, you can set the position property with any one of the following values:

- TopLeft
- TopCenter
- TopRight
- BottomLeft
- BottomCenter
- BottomRight
- LeftTop
- LeftCenter
- LeftBottom
- RightTop
- RightCenter
- RightBottom

By default, Tooltip is placed at the TopCenter of the target element.

APP.VUE

```
<template>
   <div id='app'>
        <ejs-tooltip ref="tooltip" content="Tooltip content"
position='TopCenter' target="#tooltip">
            <div id='container'>
                <ejs-button id="tooltip">Show tooltip</ejs-button>
                <div class="ddl">
                    <select id="positions" ref="positions" class="form-
control" style="width:150px" v-on:click='onclick' v-on:change='onchange'>
                        <option value="TopLeft">Top Left</option>
                        <option value="TopCenter" selected>Top
Center</option>
                        <option value="TopRight">Top Right</option>
                        <option value="BottomLeft">Bottom Left</option>
                        <option value="BottomCenter">Bottom Center</option>
                        <option value="BottomRight">Bottom Right</option>
                        <option value="LeftTop">Left Top</option>
                        <option value="LeftCenter">Left Center</option>
                        <option value="LeftBottom">Left Bottom</option>
                        <option value="RightTop">Right Top</option>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                        <option value="RightCenter">Right Center</option>
                        <option value="RightBottom">Right Bottom</option>
                    </select>
                </div>
            </div>
        </ejs-tooltip>
    </div>
</template>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(TooltipPlugin);
Vue.use(ButtonPlugin);
export default {
    data() {
    return {
    };
    },
    methods: {
        onclick: function(){
            this.$refs.tooltip.close();
        },
        onchange: function(){
            this.$refs.tooltip.position = this.$refs.positions.value;
            this.$refs.tooltip.dataBind();
        }
    }
}
</script>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#container {
    display: inline-block;
    position: relative;
    left: 50%;
    margin-top: 100px;
    transform: translateX(-50%);
    height: 200px;
}
#tooltip {
    display: inline-block;
    margin: 30px 130px 30px 50px;
}
.ddl {
    display: inline-block;
    margin: 0 30px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/tooltip/position/default-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Tip pointer positioning

The Tooltip pointer can be attached or detached from the Tooltip by using the showTipPointer property. Pointer positions can be adjusted using the tipPointerPosition property that can be assigned to one of the following values:

- Auto
- Start
- Middle
- End

The following code example illustrates how to set the pointer to the start position of the Tooltip.

APP.VUE

```
<template>
   <div id='app'>
        <ejs-tooltip content="Tooltip content" tipPointerPosition="Start"
target="#target">
            <ejs-button id="target">Show tooltip</ejs-button>
        </ejs-tooltip>
    </div>
</template>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(TooltipPlugin);
Vue.use(ButtonPlugin);
export default {
    data() {
    return {
    };
    }
}
</script>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#target {
  position: relative;
  left: 50%;
  margin-top: 100px;
  transform: translateX(-50%)
}
</style>
```

{% previewsample "page.domainurl/code-snippet/tooltip/position/tip-position-cs1" %}

By default, tip pointers are auto adjusted so that the arrow does not point outside the target element.

Dynamic positioning

The Tooltip and its tip pointer can be positioned dynamically based on the target's location. This can be achieved by using the refresh

Copyright © 2001 -2024 Syncfusion Inc.

method, which auto adjusts the Tooltip over the target.

APP.VUE

```
<template>
  <div id='app'>
        <ejs-tooltip id="targetContainer" ref="tooltip" content='Drag me
!!!' target='#demoSmart' :animation='tooltipAnimation'>
            <div id="demoSmart" ref="demoSmart"></div>
       </ejs-tooltip>
    </div>
</template>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
import { Draggable } from '@syncfusion/ej2-base';
Vue.use(TooltipPlugin);
export default {
    data() {
        return {
            tooltipAnimation: {
                open: { effect: 'None' },
                close: { effect: 'None' }
            }
        };
    },
    mounted: function (args) {
        var ele = this.$refs.demoSmart;
        var drag = new Draggable(ele, {
            clone : false,
            dragArea: '#targetContainer',
            drag: (args) => {
                if (args.element.getAttribute('data-tooltip-id') === null) {
                    this.$refs.tooltip.open(args.element);
                } else {
                    this.$refs.tooltip.refresh(args.element);
                }
            },
            dragStart: (args) => {
                if (args.element.getAttribute('data-tooltip-id') === null) {
                    this.$refs.tooltip.open(args.element);
                }
            },
            dragStop: (args) => {
                this.$refs.tooltip.close();
            }
        });
    }
}
</script>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  #app {
    height: 96%;
    position: absolute;
    width: 98%;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  }
#targetContainer {
    width: 100%;
    height: 100%;
    }
#demoSmart {
    background: rebeccapurple;
    height: 50px;
    left: 35%;
    position: absolute;
    top: 35%;
    width: 50px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/tooltip/position/dynamic-position-cs1" %}

Mouse trailing

Tooltips can be positioned relative to the mouse pointer. This behavior can be enabled or disabled by using the `mouseTrail` property. By default, it is set to `false`.

APP.VUE

```
<template>
  <div id='app'>
        <ejs-tooltip  mouseTrail='true' content='Tooltip content'
target="#target" :showTipPointer=false>
          <div id="target">Show tooltip</div>
        </ejs-tooltip>
    </div>
</template>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
Vue.use(TooltipPlugin);
export default {
    data() {
        return {};
    }
}
</script>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#target {
    background-color: #cfd8dc;
    border: 3px solid #eceff1;
    box-sizing: border-box;
    margin: 80px auto;
    padding: 20px 0;
    width: 200px;
    text-align: center;
    color: #424242;
    font-size: 20px;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/tooltip/position/mouse-trail-cs1" %}

When mouse trailing option is enabled, the tip pointer position gets auto adjusted based on the target, and

other position values like start, end, and middle are not applied (to prevent the pointer from moving out of target).

Setting offset values

Offset values are set to specify the distance between the target and tooltip element.

offsetX and offsetY properties are used to specify the offset left and top values.

- offsetX specifies the distance between the target and Tooltip element in X axis.
- offsetY specifies the distance between the target and Tooltip element in Y axis.

The following code example illustrates how to set offset values.

APP.VUE

```
<template>
  <div id='app'>
        <ejs-tooltip mouseTrail='true' content='Tooltip content'
:offsetX='xvalue' :offsetY='yvalue' :showTipPointer=false target="#target">
          <div id="target">Show tooltip</div>
        </ejs-tooltip>
    </div>
</template>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
Vue.use(TooltipPlugin);
export default {
    data() {
    return {
        xvalue: 30,
        yvalue: 30,
        tip: false
    };
    }
}
</script>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#target {
    background-color: #cfd8dc;
    border: 3px solid #eceff1;
    box-sizing: border-box;
    margin: 80px auto;
    padding: 20px 0;
    width: 200px;
    text-align: center;
    color: #424242;
    font-size: 20px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</style>
```

{% previewsample "page.domainurl/code-snippet/tooltip/position/offset-value-cs1" %}

By default, collision is handled automatically and therefore when collision is detected the Tooltip fits horizontally and flips vertically.

Open mode in Vue Tooltip component

You can decide the mode on which the Tooltip is to be opened on a page, i.e., on hovering, focusing, or clicking on the target elements.

On mobile devices, Tooltips appear when you tap and hold the element, even if the opensOn option is assigned with hover.

Tooltips are also displayed as long as you continue to tap and hold the element. On lift, it disappears in the next 1.5 seconds.

If there is another action before that time ends, then the Tooltip disappears.

The opensOn property can take either a single or a combination of multiple values, separated by space from the following options. The table below explains how the Tooltip opens on both desktop and mobile based on the value(s) assigned to the opensOn property. By default, it takes auto value.

| Values | Desktop | Mobile |

| ---------- | ---------- | ---------- |

| Auto | Tooltip appears when you hover over the target or when the target element receives the focus. | Tooltip opens on tap and hold of the target element. |

| Hover | Tooltip appears when you hover over the target. | Tooltip opens on tap and hold of the target element. |

| Click | Tooltip appears when you click a target element. | Tooltip appears when you single tap the target element. |

| Focus | Tooltip appears when you focus (say through tab key) on a target element. | Tooltip appears with a single tap on the target element. |

| Custom | Tooltip is not triggered by any default action. So, you have to bind your own events and use either open or close public methods. | Same as Desktop. |

To open the Tooltip for multiple actions, say while hovering over or clicking on a target element, the opensOn property can be assigned with multiple values, separated by space as Hover Click.

Auto value cannot be used with any combination for multiple values.

The following code example shows how to set the open mode for Tooltips.

APP.VUE

```
<template>
   <div id='app'>
        <div id='container'>
            <ejs-tooltip id="tooltipHover" opensOn='Hover' content='Tooltip
from hover' target="#tooltiphover">
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <ejs-tooltip target="#tooltipclick" id="tooltipClick"
opensOn='Click' content='Tooltip from click'>
                    <ejs-tooltip target="#tooltipfocus" id="tooltipFocus"
opensOn='Focus' content='Tooltip from focus'>
                        <ejs-tooltip target="#tooltipopen" ref="tooltip"
opensOn='Custom' content='Tooltip from custom mode'>
                            <table style="margin-top: 50px;transform:
translateX(-50%);position: relative;left: 50%;height: 250px;">
                                <tbody>
                                    <tr>
                                        <td style="padding:25px">
                                            <div id="tooltiphover"
class="blocks e-btn">
                                                <span>Hover Me
!(Default)</span>
                                            </div>
                                        </td>
                                        <td style="padding:25px">
                                            <div id="tooltipclick"
class="blocks e-btn">
                                                <span>Click Me !</span>
                                            </div>
                                        </td>
                                    </tr>
                                    <tr>
                                        <td style="padding:25px">
                                            <div class="">
                                                <span id="textbox" class="e-
float-input">
                                                    <input id="tooltipfocus"
type="text" placeholder="Focus and blur" />
                                                </span>
                                            </div>
                                        </td>
                                        <td style="padding:25px">
                                            <div id="tooltipcustom">
                                                <div>
                                                    <input id="tooltipopen"
type="button" class="e-btn" value="Click to open tooltip manually" v-
on:click="onCustomOpen" />
                                                </div>
                                            </div>
                                        </td>
                                    </tr>
                                </tbody>
                            </table>
                        </ejs-tooltip>
                    </ejs-tooltip>
                </ejs-tooltip>
            </ejs-tooltip>
        </div>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
.blocks {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    width: 260px;
}
</style>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(TooltipPlugin);
Vue.use(ButtonPlugin);
export default {
  data: function() {
    return {
    };
  },
  methods: {
        onCustomOpen: function(args){
           if (args.target.getAttribute('data-tooltip-id')) {
           this.$refs.tooltip.close();
        } else {
            this.$refs.tooltip.open(args.target);
        }
        }
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/open-mode-cs1" %}

Custom open mode

Other than the above specified options, the custom mode makes the Tooltip appear on screen for user-defined custom actions such as

right-click, double-click, and so on. Here, the tooltip is not triggered by any default action, and you have to bind your own events and use either open or close public methods to show or hide the Tooltips.

The following code example shows how to define custom open mode for the Tooltip.

APP.VUE

```
<template>
   <div id='app'>
    <ejs-tooltip ref="tooltip" id="tooltip" content='Tooltip from custom
mode' opensOn='Custom' v-on:dblclick.native='customOpen'>
            <div id="box">Double-click to open Tooltip.</div>
    </ejs-tooltip>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#box {
    background-color: #cfd8dc;
    border: 3px solid #eceff1;
    box-sizing: border-box;
    margin: 80px auto;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    padding: 20px 0;
    width: 300px;
    text-align: center;
    color: #424242;
    font-size: 20px;
    user-select: none;
}
</style>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
Vue.use(TooltipPlugin);
export default {
  data: function() {
    return {
    };
  },
  methods: {
        customOpen: function(args){
           if (args.target.getAttribute('data-tooltip-id')) {
           this.$refs.tooltip.close();
        } else {
            this.$refs.tooltip.open(args.target);
        }
        }
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/custom-open-cs1" %}

Sticky mode

With this mode set to true, Tooltips can be made to show up on the screen as long as the close icon is pressed. In this mode, close icon is attached to the Tooltip located at the top right corner. This mode can be enabled or disabled using the isSticky property.

APP.VUE

```
<template>
   <div id='app'>
        <ejs-tooltip id="tooltip" content='Click close icon to close me'
isSticky='true'>
            <div id="target">
                Show tooltip
            </div>
    </ejs-tooltip>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#target {
    background-color: #cfd8dc;
    border: 3px solid #eceff1;
    box-sizing: border-box;
    margin: 80px auto;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    padding: 20px 0;
    width: 200px;
    text-align: center;
    color: #424242;
    font-size: 20px;
    user-select: none;
}
</style>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
Vue.use(TooltipPlugin);
export default {
  data: function() {
    return {
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/sticky-cs1" %}

Open/Close Tooltip with delay

The Tooltips can be opened or closed after some delay by using the openDelay and closeDelay properties.

APP.VUE

```
<template>
   <div id='app'>
        <ejs-tooltip id="tooltip" content='Tooltip with delay'
openDelay='1000' closeDelay='1000'>
                <div id="target">
                    Show tooltip
                </div>
        </ejs-tooltip>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#target {
    background-color: #cfd8dc;
    border: 3px solid #eceff1;
    box-sizing: border-box;
    margin: 80px auto;
    padding: 20px 0;
    width: 200px;
    text-align: center;
    color: #424242;
    font-size: 20px;
    user-select: none;
}
</style>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
Vue.use(TooltipPlugin);
export default {
  data: function() {
    return {
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/open-close-cs1" %}

Animation in Vue Tooltip component

To animate the Tooltip, a set of specific animation effects are available, and it can be controlled using the animation property.The animation property also allows you to set delay, duration, and various other effects of your choice.

AnimationModel is derived from base to apply the chosen animation effect, duration, etc. on Tooltips.

By default, Tooltip entrance occurs over 150 ms using the ease-out timing function. It exits also at 150 ms, but uses ease-in timing function.

The default animation effect for the Tooltip is set to FadeIn for its open action, and FadeOut for its close action. The default duration is set to 150 ms and delay is set to 0.

APP.VUE

```
<template>
   <div id='app'>
      <ejs-tooltip id="tooltip" ref="tooltip" content='Tooltip animation
effect' :animation='TooltipAnimation'>
            <div id="target">
                Show tooltip
            </div>
    </ejs-tooltip>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#target {
    background-color: #cfd8dc;
    border: 3px solid #eceff1;
    box-sizing: border-box;
    margin: 80px auto;
    padding: 20px 0;
    width: 200px;
    text-align: center;
    color: #424242;
    font-size: 20px;
    user-select: none;
}
</style>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
```

```
Vue.use(TooltipPlugin);
export default {
  data: function() {
    return {
      TooltipAnimation: {
        open: { effect: 'ZoomIn', duration: 1000, delay: 0 },
        close: { effect: 'ZoomOut', duration: 500, delay: 0 }
    }
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/animation/default-animation-cs1" %}

Animation effects

The animation effects that are applicable to Tooltips are:

- FadeIn
- FadeOut
- FadeZoomIn
- FadeZoomOut
- FlipXDownIn
- FlipXDownOut
- FlipXUpIn
- FlipXUpOut
- FlipYLeftIn
- FlipYLeftOut
- FlipYRightIn
- FlipYRightOut
- ZoomIn
- ZoomOut
- None

When the `effect` is specified as `none`, no effect will be applied to the Tooltip, and animation is considered to be set to `off`.

Some of the above animation effects suits the Tooltip better when its tip pointer is hidden.

This can be achieved by setting the `showTipPointer` to false.

Animating via open/close methods

Animations can also be applied on Tooltips dynamically through `open` and `close` methods. These methods accept the animation model as an

optional parameter. If you pass `TooltipAnimationSettings`, animation takes this model; otherwise, it takes the values from the

`animation` property. It is also possible to pass different animations for Tooltips on each target.

Refer to the code snippet below to apply animations using public methods.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
   <div id='app'>
     <ejs-tooltip id="tooltip" ref="tooltip" content='Tooltip animation
effect' opensOn='Custom' v-on:click.native='onCustomClick'>
            <div id="target">
                Show tooltip
            </div>
    </ejs-tooltip>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#target {
    background-color: #cfd8dc;
    border: 3px solid #eceff1;
    box-sizing: border-box;
    margin: 80px auto;
    padding: 20px 0;
    width: 200px;
    text-align: center;
    color: #424242;
    font-size: 20px;
    user-select: none;
}
</style>
<script>
import Vue from "vue";
import { TooltipPlugin, TooltipAnimationSettings } from "@syncfusion/ej2-
vue-popups";
Vue.use(TooltipPlugin);
export default {
  data: function() {
    return {
    };
  },
  methods: {
        onCustomClick: function(args){
          if (args.target.getAttribute("data-tooltip-id")) {
        let closeAnimation = { effect: 'FadeOut', duration: 1000 }
        this.$refs.tooltip.close(closeAnimation);
    } else {
        let openAnimation = { effect: 'FadeIn', duration: 1000 }
        this.$refs.tooltip.open(this.$refs.tooltip.$el, openAnimation);
    }
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/animation/animation-effects-cs1" %}

Apply transition

The transition effect can be applied on Tooltips by using the `beforeRender` event as given in the following work-around code example.

APP.VUE

```
<template>
   <div id='app'>
       <h3 style="text-align: center;"> Transition effect </h3>
    <ejs-tooltip id="tooltip" ref="tooltip" class="e-prevent-select"
target='.circle-tool' closeDelay='1000' :animation='Animation'
     :beforeRender='onBeforeRender' :beforeOpen='onBeforeOpen'
:afterClose='onAfterClose'>
        <div class="circle-tool" style="top:18%;left:5%" title="This is
Turtle !!!"></div>
        <div class="circle-tool" style="top:30%;left:30%" title="This is
Snake !!!"></div>
        <div class="circle-tool" style="top:80%;left:80%" title="This is
Croc !!!"></div>
        <div class="circle-tool" style="top:65%;left:50%" title="This is
String Ray !!!"></div>
        <div class="circle-tool" style="top:75%;left:15%" title="This is
Blob Fish !!!"></div>
        <div class="circle-tool" style="top:30%;left:70%" title="This is
Mammoth !!!"></div>
    </ejs-tooltip>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
   #app {
    height: 40px;
    position: absolute;
    width: 95%;
  }
   #tooltip {
        display: block;
        border: 1px solid #c8c8c8;
        height: 200px;
        margin-left: 10px;
        margin-right: 10px;
        position: relative;
    }
    .circle-tool {
        position: absolute;
        width: 20px;
        height: 20px;
        background: #9acd32;
        -moz-border-radius: 50px;
        -webkit-border-radius: 50px;
        border-radius: 50px;
    }
</style>
<script>
import Vue from "vue";
import { TooltipPlugin, TooltipEventArgs  } from "@syncfusion/ej2-vue-
popups";
Vue.use(TooltipPlugin);
export default {
  data: function() {
    return {
      Animation: {
```

```
        open: { effect: 'ZoomIn', duration: 500 },
        close: { effect: 'ZoomOut', duration: 500 }
    }
    };
  },
  methods: {
        onBeforeRender: function(args) {
        if (args.element) {
            this.$refs.tooltip.animation = { open: { effect: 'None' } };
        }
    }
    onBeforeOpen: function(args){
        if (args.element) {
            args.element.style.display = 'block';
            args.element.style.transitionProperty = 'left,top';
            args.element.style.transitionDuration = '1000ms';
        }
    }
    onAfterClose: function(args) {
        this.$refs.tooltip.animation = {
            open: { effect: 'ZoomIn', duration: 500 },
            close: { effect: 'ZoomOut', duration: 500 }
        };
    }
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/animation/transition-cs1" %}

Customization in Vue Tooltip component

The Tooltip can be customized by using the cssClass property, which accepts custom CSS class names that define specific user-defined styles and themes to be applied on the Tooltip element.

Tip pointer customization

Styling the tip pointer's size, background, and border colors can be done using the cssClass property, as given below.

APP.VUE

```
<template>
    <div id="app">
    <ejs-tooltip ref="tooltip"  class="tooltipcontainer"
:cssClass="cssClass" content='Tooltip arrow customized' >
            <div id="target">
                Show tooltip
            </div>
    </ejs-tooltip>
  </div>
</template>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
Vue.use(TooltipPlugin);
export default {
    data() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    return {
        cssClass: 'customtip'
    }
    }
}
</script>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#target {
    background-color: #cfd8dc;
    border: 3px solid #eceff1;
    box-sizing: border-box;
    margin: 80px auto;
    padding: 20px 0;
    width: 200px;
    text-align: center;
    color: #424242;
    font-size: 20px;
    user-select: none;
}
/* csslint ignore:start */
.customtip.e-tooltip-wrap {
    padding: 4px;
}
.customtip.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
    height: 20px;
    width: 12px;
}
.customtip.e-tooltip-wrap .e-arrow-tip.e-tip-top {
    height: 20px;
    width: 12px;
}
.customtip.e-tooltip-wrap .e-arrow-tip.e-tip-left {
    height: 12px;
    width: 20px;
}
.customtip.e-tooltip-wrap .e-arrow-tip.e-tip-right {
    height: 12px;
    width: 20px;
}
.customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
    border-left: 6px solid transparent;
    border-right: 6px solid transparent;
    border-top: 20px solid #616161;
}
.customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
    border-bottom: 20px solid #616161;
    border-left: 6px solid transparent;
    border-right: 6px solid transparent;
}
.customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
    border-bottom: 6px solid transparent;
    border-right: 20px solid #616161;
    border-top: 6px solid transparent;
}
.customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    border-bottom: 6px solid transparent;
    border-left: 20px solid #616161;
    border-top: 6px solid transparent;
}
/* csslint ignore:end */
</style>
```

{% previewsample "page.domainurl/code-snippet/tooltip/custom-tip-cs1" %}

Tooltip customization

The complete look and feel of the Tooltip can be customized by changing it's background color, opacity, content font, etc.

The following code example shows the way to achieve it.

APP.VUE

```
<template>
    <div id="app">
    <ejs-tooltip ref="tooltip"  class="tooltipcontainer"
:cssClass="cssClass" content='Tooltip arrow customized' >
            <div id="target">
                Show tooltip
            </div>
        </ejs-tooltip>
  </div>
</template>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
Vue.use(TooltipPlugin);
export default {
    data() {
    return {
        cssClass: 'customtooltip'
    }
    }
}
</script>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#target {
    background-color: #cfd8dc;
    border: 3px solid #eceff1;
    box-sizing: border-box;
    margin: 80px auto;
    padding: 20px 0;
    width: 200px;
    text-align: center;
    color: #424242;
    font-size: 20px;
    user-select: none;
}
/* csslint ignore:start */
.customtooltip.e-tooltip-wrap .e-tip-content {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    line-height: 20px;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
    height: 12px;
    left: 50%;
    top: 100%;
    width: 24px;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip.e-tip-top {
    height: 12px;
    left: 50%;
    top: -9px;
    width: 24px;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip.e-tip-left {
    height: 24px;
    left: -9px;
    top: 48%;
    width: 12px;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip.e-tip-right {
    height: 24px;
    left: 100%;
    top: 50%;
    width: 12px;
}
.customtooltip.e-tooltip-wrap {
    border-radius: 4px;
    opacity: 1;
}
.customtooltip.e-tooltip-wrap.e-popup {
    background-color: #fff;
    border: 2px solid #000;
}
.customtooltip.e-tooltip-wrap .e-tip-content {
    color: #000;
    font-size: 12px;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
    border-left: 12px solid transparent;
    border-right: 14px solid transparent;
    border-top: 12px solid #000;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
    border-bottom: 12px solid #000;
    border-left: 12px solid transparent;
    border-right: 12px solid transparent;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
    border-bottom: 12px solid transparent;
    border-right: 12px solid #000;
    border-top: 12px solid transparent;
}
.customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
    border-bottom: 12px solid transparent;
    border-left: 12px solid #000;
    border-top: 12px solid transparent;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
.customtooltip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {
    color: #fff;
    font-size: 25.9px;
}
/* csslint ignore:end */
</style>
```

{% previewsample "page.domainurl/code-snippet/tooltip/custom-tooltip-cs1" %}

Style in Vue Tooltip component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the tooltip

Use the following CSS to customize the tooltip.

`

.e-tooltip-wrap {

border-radius: 4px;

opacity: 1;

}

`

Customizing the tooltip popup

Use the following CSS to customize the tooltip popup properties.

`

.e-tooltip-wrap.e-popup {

background-color: #fff;

border: 2px solid #000;

}

`

Customizing the tooltip content

Use the following CSS to customize the tooltip content.

`

.e-tooltip-wrap .e-tip-content {

color: red;

font-size: 12px;

line-height: 20px;

}

`

Copyright © 2001 -2024 Syncfusion Inc.

Customizing the tooltip arrow tip

Use the following CSS to customize the tooltip arrow tip.

```
/ To customize the arrow tip at bottom /
.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
height: 12px;
left: 50%;
top: 100%;
width: 24px;
}
/ To customize the arrow tip at top /
.e-tooltip-wrap .e-arrow-tip.e-tip-top {
height: 12px;
left: 50%;
top: -9px;
width: 24px;
}
/ To customize the arrow tip at left /
.e-tooltip-wrap .e-arrow-tip.e-tip-left {
height: 24px;
left: -9px;
top: 48%;
width: 12px;
}
/ To customize the arrow tip at right /
.e-tooltip-wrap .e-arrow-tip.e-tip-right {
height: 24px;
left: 100%;
top: 50%;
width: 12px;
}
```

Customizing the tooltip inner tip

Use the following CSS to customize the tooltip inner tip.

Copyright © 2001 -2024 Syncfusion Inc.

```
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-right,
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-left,
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom,
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-top {
color: #fff;
font-size: 25.9px;
}
```

Customizing the tooltip outer tip

Use the following CSS to customize the tooltip outer tip.

```
/ To customize the arrow tip at bottom /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
border-left: 12px solid transparent;
border-right: 14px solid transparent;
border-top: 12px solid #000;
}
/ To customize the arrow tip at top /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
border-bottom: 12px solid #000;
border-left: 12px solid transparent;
border-right: 12px solid transparent;
}
/ To customize the arrow tip at left /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
border-bottom: 12px solid transparent;
border-right: 12px solid #000;
border-top: 12px solid transparent;
}
/ To customize the arrow tip at right /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
border-bottom: 12px solid transparent;
```

Copyright © 2001 -2024 Syncfusion Inc.

border-left: 12px solid #000;

border-top: 12px solid transparent;

}

`

Accessibility in Vue Tooltip component

The Tooltip component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Tooltip component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

Copyright © 2001 -2024 Syncfusion Inc.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Tooltip component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Tooltip component.

| Attributes | Description |

| --- | --- |

| role="tooltip" | The element that serves as the container for the tooltip has the ARIA role of tooltip. |

| aria-describedby | This attribute is added to the target element on which the Tooltip gets opened, when focusing or hovering over it. It usually holds the randomly generated Id value of the Tooltip element.

In case, the target element already holds an aria-describedby attribute with Id value of some other component, then the Tooltip Id value can be additionally appended to the existing aria-describedby attribute separated by a space as shown in the example below.

 For example:
 aria-describedby = "my-text my-tooltip"
 my-text is the Id of some other component.
 my-tooltip is the id of Tooltip component.

 When the Tooltip is closed, the aria-describedby attribute is removed from the target. |

| aria-hidden | This attribute is assigned to the Tooltip element whose default value is true.

 When true, it denotes that the Tooltip element is in a hidden or a closed state. When the Tooltip appears on the screen, it’s value changes to false.|

Keyboard interaction

The Tooltip component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Tooltip component.

| Keys | Description |

| --- | --- |

| Escape | Closes or dismisses the Tooltip. |

| Tab | A form control receiving focus (say through tab key), opens the Tooltip, and on focus out closes it. |

1. When the Tooltip is being displayed on the target element, focus continues to stay on it.

2. If the Tooltip opens on mouse entering into the target element space, then it should be dismissed only when the mouse leaves that target.

3. If the Tooltip opens on the target element that receives focus, then it should be closed only when the focus moves out of that target element.

Likewise, if the Tooltip opens on a click, then it should be closed only on another click action.

Ensuring accessibility

The Tooltip component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

Copyright © 2001 -2024 Syncfusion Inc.

The accessibility compliance of the Tooltip component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Tooltip component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/tooltip.html" %}

See also

- Accessibility in Syncfusion Vue components

How To

Show tooltip on disabled elements and disable tooltip in Vue Tooltip component

By default, tooltips will not be displayed on disabled elements. However, you can enable this behavior by using the following steps:

1. Add a disabled element like the button element into a div whose display style is set to inline-block.
2. Set the pointer event as none for the disabled element (button) through CSS.
3. Initialize the tooltip for outer div element that holds the disabled button element.

APP.VUE

```
<template>
   <div id='app'>
    <ejs-tooltip id='tooltip' :content='content' target='#box'>
        <div id='container'>
            <div id="box" style="display: inline-block;">
                <ejs-button id='disabledbutton' disabled=true>Disabled
button</ejs-button>
            </div>
        </div>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#disabledbutton {
    pointer-events: none;
    font-size: 22px;
    padding: 10px;
}
#container {
    display: inline-block;
    position: relative;
    left: 50%;
    transform: translateX(-50%);
    margin-top: 75px;
}
</style>
<script>
import Vue from "vue";
import { TooltipPlugin } from "@syncfusion/ej2-vue-popups";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(TooltipPlugin);
Vue.use(ButtonPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data: function() {
    return {
        content: "Tooltip from disabled element"
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/how-to-cs1" %}

Dynamic tooltip content with html elements in Vue Tooltip component

The Tooltip component loads HTML tags using the content template.

The HTML tags such as , , bold, italic, underline, etc., can be used. Style attributes can also be applied with HTML tags.

Here, Bold, Italic, Underline, and Anchor tags are used.

APP.VUE

```
<template>
   <div id='app'>
        <ejs-tooltip target='#title' cssClass='e-tooltip-css'
ref="tooltipTitle" position='BottomCenter' opensOn='Click' ref="tooltip"
:content="content">
            <div id="container">
                <div id="tooltipContent">
                    <div class="content">
                        <ejs-button class="text" id="title">HTML(With
Title)</ejs-button>
                    </div>
                </div>
            </div>
        </ejs-tooltip>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#tooltipContent table {
    margin: 0 auto;
}
#tooltipContent {
    display: inline-block;
    position: relative;
    left: 50%;
    transform: translateX(-50%);
    margin-top: 50px;
    height: 200px;
}
.text {
    text-transform: capitalize;
    width: 155px;
}
.header {
    font-family: "Arial Black", Gadget, sans-serif;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    font-size: 12px;
    padding-bottom: 2px;
    margin: 4px -7px 7px -7px;
    padding-right: 5px;
    padding-left: 6px;
    font-weight: bold;
    height: 18px;
    border-bottom: 1px solid white;
}
.e-tooltip-css.e-tooltip-wrap .e-tip-content {
    padding: 0 10px 10px 10px;
}
</style>
<script>
import Vue from 'vue';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { TooltipPlugin } from '@syncfusion/ej2-vue-popups';
Vue.use(ButtonPlugin);
Vue.use(TooltipPlugin);
var demoVue = Vue.component("demo", {
  template: `
    <div id="tooltip" ref="content">
        <h2>HTML Tags</h2>
            Through templates, <b><span style="color:#e3165b">tooltip
content</span></b> can be loaded with <u><i> inline HTML, images, iframe,
videos, maps </i></u>. A title can be added to the content
    </div>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
  data: function() {
    return {
        content: function () {
            return { template : demoVue}
        }
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/tags-cs1" %}

Custom tooltip with dynamic html in Vue Tooltip component

Tooltip loads HTML pages via HTML tags such as iframe, video, and map using the content property, which supports both string and HTML tags.

To load an iframe element in tooltip, set the required iframe in the content of tooltip while initializing the tooltip component. Refer to the following code.

`ts

Copyright © 2001 -2024 Syncfusion Inc.

content= '<iframe
src="https://ej2.syncfusion.com/showcase/typescript/expensetracker/#/dashboard"></iframe>
`

APP.VUE

```
<template>
   <div id='app'>
        <ejs-tooltip target='#iframeContent' cssClass='e-tooltip-css'
ref="tooltipTitle" :position='position' :opensOn='opensOn'
:content='content'>
            <div id='container'>
                <div id="tooltipContent">
                    <div class="content">
                        <ejs-button cssClass="text"
id="iframeContent">Embedded Iframe</ejs-button>
                    </div>
                </div>
            </div>
        </ejs-tooltip>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
</style>
<script>
import Vue from 'vue';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ButtonPlugin);
import { TooltipPlugin } from '@syncfusion/ej2-vue-popups';
Vue.use(TooltipPlugin);
export default {
  data: function() {
    return {
      position: 'BottomCenter',
      opensOn: 'Click',
      content: '<iframe
src="https://ej2.syncfusion.com/showcase/typescript/expensetracker/#/dashboa
rd"></iframe>'
    };
  },
  methods: {
  }
}
</script>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#tooltipContent {
    position: relative;
    left: 50%;
    transform: translateX(-50%);
    margin: 65px 10px 0 10px;
}
.content {
```

```
    display: inline-block;
    width: 49%;
    height: 250px;
    position: relative;
    left: 50%;
    transform: translateX(-50%);
}
#tooltipFrame {
    width: 332px;
    height: 233px;
}
.content .e-btn {
    position: relative;
    left: 50%;
    transform: translateX(-50%);
}
</style>
```

{% previewsample "page.domainurl/code-snippet/tooltip/iframe-cs1" %}

Tooltip open or display modes in Vue Tooltip component

The open mode property of tooltip can be defined on a target that is hovering, focusing, or clicking.

Tooltip component have the following types of open mode:

- Auto
- Hover
- Click
- Focus
- Custom

Auto

Tooltip appears when you hover over the target or when the target element receives the focus.

Hover

Tooltip appears when you hover over the target.

Click

Tooltip appears when you click a target element.

Focus

Tooltip appears when you focus (say through tab key) on a target element.

Custom

Tooltip is not triggered by any default action. So, bind your own events and use either open or close public methods.

APP.VUE

```
<template>
   <div id='app'>
       <div id='container'>
           <ejs-tooltip cssClass='e-tooltip-css' :opensOn='opensOnHover'
target='#tooltiphover' :content='hoverContent'>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <ejs-tooltip cssClass='e-tooltip-css'
:opensOn='opensOnClick' target='#tooltipclick' :content='clickContent'>
                    <ejs-tooltip cssClass='e-tooltip-css' opensOn="Click"
isSticky=true target='#Mode' :content='stickyContent'>
                        <ejs-tooltip cssClass='e-tooltip-css'
target='#openMode' openDelay=1000 closeDelay=1000 :content='delayContent'>
                            <ejs-tooltip cssClass='e-tooltip-css'
:opensOn='opensOnCustom' target='#tooltipcustom' id='custom' ref='custom'
:content='customContent'>
                                <ejs-tooltip cssClass='e-tooltip-css'
:opensOn='opensOnFocus' target='#focus' :content='focusContent'>
                                    <div id="showTooltip">
                                        <div id="first">
                                            <ejs-button class="blocks e-
outline" isPrimary=true id="tooltiphover">Hover me!(Default)</ejs-button>
                                            <ejs-button class="blocks e-
outline" isPrimary=true id="tooltipclick">Click me!</ejs-button>
                                        </div>
                                        <br/><br/>
                                        <div id="second">
                                            <ejs-button class="blocks e-
outline" isPrimary=true id="Mode">Sticky Mode</ejs-button>
                                            <ejs-button class="blocks e-
outline" isPrimary=true id="openMode">Tooltip with delay</ejs-button>
                                        </div>
                                        <br/><br/>
                                        <div id="third">
                                        <button class="blocks e-outline e-
btn e-primary" v-on:dblclick="doubleClick" id="tooltipcustom" >Double click
on me!</button>
                                            <div id="textbox" class="e-
float-input blocks">
                                                <input id="focus"
type="text" placeholder="Focus and blur"/>
                                            </div>
                                        </div>
                                    </div>
                                </ejs-tooltip>
                            </ejs-tooltip>
                        </ejs-tooltip>
                    </ejs-tooltip>
                </ejs-tooltip>
            </ejs-tooltip>
        </div>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#container {
  width: 100%;
}
#textbox {
  display: inline-block;
  top: -3px;
}
.blocks {
```

```
  margin: 0 10px 0 10px;
  text-transform: none;
  width: 168px;
}
#showTooltip {
  display: table;
  padding-top: 40px;
  margin: 0 auto;
}
#focus {
  border: 1px solid #ff4081;
  text-align: center;
  height: 31px;
  width: 168px;
}
/* csslint ignore:start */
::placeholder {
  color: #ff4081;
}
/* csslint ignore:end */
</style>
<script>
import Vue from 'vue';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ButtonPlugin);
import { TooltipPlugin } from '@syncfusion/ej2-vue-popups';
Vue.use(TooltipPlugin);
export default {
  data: function() {
    return {
            position: 'BottomCenter',
            hoverContent: 'Tooltip from hover',
            stickyContent: 'Click close icon to close me',
            delayContent:  'Opens and closes Tooltip with delay of <i>1000
milliseconds</i>',
            customContent: 'Tooltip from custom mode',
            clickContent:'Tooltip from click',
            focusContent:'Tooltip from focus',
            opensOnHover: 'Hover',
            opensOnClick: 'Click',
            opensOnCustom: 'Custom',
            opensOnFocus: 'Focus',
    };
  },
  methods: {
    doubleClick(args) {
        if (args.target.getAttribute('data-tooltip-id')) {
            this.$refs.custom.close();
        } else {
            this.$refs.custom.open(args.target);
        }
    }
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/tooltip/mode-cs1" %}

Fancy tooltip customization in Vue Tooltip component

The arrow of the tooltip can be customized as needed by customizing CSS in the sample-side. The EJ2 tooltip component is achieved through CSS3 format and positioned the tip arrow according to the tooltip positions like TopCenter, BottomLeft, RightTop, and more.

Here, the tip arrow is customized as Curved tooltip and Bubble tooltip.

Curved tip

The content for the tip pointer arrow has been added. To customize the curved tip arrow, override the following CSS class of tip arrow.

```ts
.e-arrow-tip-outer.e-tip-bottom,
.e-arrow-tip-outer.e-tip-top {
border-left: none !important;
border-right: none !important;
border-top: none !important;
}
.e-arrow-tip.e-tip-top {
transform: rotate(170deg);
}
```

Bubble tip

The two divs(inner div and outer div) have been added to achieve the bubble tip arrow. To customize the bubble tip arrow, override the following CSS class of tip arrow.

```ts
.e-arrow-tip-outer.e-tip-bottom, .e-arrow-tip-outer.e-tip-top
{
border-radius: 50px;
height: 10px;
width: 10px;
}
.e-arrow-tip-inner.e-tip-bottom, .e-arrow-tip-inner.e-tip-top
{
border-radius: 50px;
height: 10px;
width: 10px;
```

Copyright © 2001 -2024 Syncfusion Inc.

}
`

These tip arrow customizations have been achieved through CSS changes in the sample level. The tooltip position can be changed by using the radio button click event.

The arrow tip pointer can also be disabled by using the showTipPointer property in a tooltip.

APP.VUE

```
<template>
   <div id='app'>
        <ejs-tooltip ref='tooltip_1' cssClass="curvetips e-tooltip-css"
content="Tooltip arrow customized" target="#target">
            <ejs-tooltip mouseTrail=true :showTipPointer=false
content='Disabled tooltip pointer' target='#tooltip' cssClass='pointertip e-
tooltip-css'>
                <ejs-tooltip position="TopRight" content='Tooltip arrow
customized as balloon tip' target='#bubbletip' cssClass='bubbletip e-
tooltip-css' ref="tooltip_2">
                    <div id='container'>
                        <div id="customization" class="customTipContainer">
                            <ejs-button id="target">
                                Customized Tip Arrow
                            </ejs-button>
                            <div id="positions">
                                <ul>
                                    <li>
                                        <ejs-radiobutton id='element1'
label='TopCenter' name='default' value='TopCenter' checked=true  v-
on:change='onChange'/>
                                    </li>
                                    <li>
                                        <ejs-radiobutton id='element2'
label='BottomLeft' name='default' value='BottomLeft' v-
on:change='onChange'/>
                                    </li>
                                </ul>
                            </div>
                        </div>
                        <div id="balloon" class="customTipContainer">
                            <ejs-button id="bubbletip">
                                Bubble Tip Arrow
                            </ejs-button>
                            <div id="btn">
                                <ul>
                                    <li>
                                        <ejs-radiobutton id='radio1'
label='BottomLeft' name='position' value='BottomLeft' v-
on:change='onChanged'/>
                                    </li>
                                    <li>
                                        <ejs-radiobutton id='radio2'
label='TopRight' name='position' value='TopRight'  checked=true v-
on:change='onChanged'/>
                                    </li>
                                </ul>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                            </div>
                        </div>
                        <div id="disabledContainer"
class="customTipContainer">
                            <ejs-button id="tooltip">
                                Disabled Tip Arrow
                            </ejs-button>
                        </div>
                    </div>
                </ejs-tooltip>
            </ejs-tooltip>
        </ejs-tooltip>
    </div>
</template>
<script>
import Vue from 'vue';
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ButtonPlugin);
import { TooltipPlugin } from '@syncfusion/ej2-vue-popups';
Vue.use(TooltipPlugin);
import { RadioButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(RadioButtonPlugin);
export default {
  data: function() {
    return {
    };
  },
  methods: {
        onChange(args) {
            this.$refs.tooltip_1.position = args.value;
            this.$refs.tooltip_1.dataBind();
        },
        onChanged(args) {
            this.$refs.tooltip_2.position = args.value;
            if(this.$refs.tooltip_2.position == 'BottomLeft'){
             this.$refs.tooltip_2.offsetY = -30;
            } else {
             this.$refs.tooltip_2.offsetY = 0;
            }
             this.$refs.tooltip_2.dataBind();
        }
  }
}
</script>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#bubbletip {
  border-color: #d2a679;
}
#target {
  border-color: #e86238;
}
li {
  list-style: none;
}
.e-radio-wrapper {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  margin-top: 18px;
}
#target,
#bubbletip,
#tooltip {
  box-sizing: border-box;
  padding: 20px;
  width: 200px;
  text-align: center;
  top: -17px;
  margin-bottom: 40px;
}
/* csslint ignore:start */
@font-face {
  font-family: "tip";
  src:
url("https://ej2.syncfusion.com/products/typescript/tooltip/customization/Fo
nts/tip.tff") format("truetype"),
url("https://ej2.syncfusion.com/products/typescript/tooltip/customization/Fo
nts/tip.woff") format("woff"),
url("https://ej2.syncfusion.com/products/typescript/tooltip/customization/Fo
nts/tip.eot") format("eot"),
url("https://ej2.syncfusion.com/products/typescript/tooltip/customization/Fo
nts/tip.svg?#tip") format("svg");
  font-weight: normal;
  font-style: normal;
}
/* csslint ignore:end */
/* csslint ignore:start */
#container {
  width: 100%;
}
.customTipContainer {
  width: 400px;
  position: relative;
  left: 50%;
  transform: translateX(-50%);
  top: 50px;
}
#disabledContainer {
  margin-top: 25px;
}
.pointertip.e-tooltip-wrap .e-tip-content,
.curvetips.e-tooltip-wrap .e-tip-content {
  color: white;
}
.pointertip.e-tooltip-wrap.e-popup {
  background-color: #80180d;
  border: 3px solid #ff9999;
}
.curvetips .e-arrow-tip.e-tip-top {
  margin-left: -20px;
  top: -16px;
  transform: rotate(177deg);
  left: 50px;
}
.curvetips.e-tooltip-wrap {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  padding: 17px;
  border-radius: 5px;
}
.curvetips.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom:before,
.curvetips.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top:before {
  font-family: "tip" !important;
  speak-as: none;
  font-size: 21px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  content: "\e700";
  color: #e86238;
}
.curvetips.e-tooltip-wrap.e-popup {
  background: #e86238;
  border: none;
}
.curvetips.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom,
.curvetips.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
  border-left: none;
  border-right: none;
  border-top: none;
}
.curvetips.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom:before,
.curvetips.e-tooltip-wrap .e-arrow-tip-inner.e-tip-top:before {
  content: none;
}
.curvetips.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom,
.curvetips.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
  top: -1px;
}
.curvetips.e-tooltip-wrap .e-arrow-tip.e-tip-bottom,
.curvetips.e-tooltip-wrap .e-arrow-tip.e-tip-top {
  position: absolute;
  height: 18px;
  width: 28px;
}
#positions {
  display: inline-block;
}
#btn {
  display: inline-block;
}
#target .e-tip-content {
  padding: 0px;
}
.bubbletip.e-tooltip-wrap {
  padding: 8px;
}
.bubbletip.e-tooltip-wrap .e-tip-content {
  border-radius: 50%;
  text-align: center;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  color: white;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
  height: 40px;
  width: 50px;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip.e-tip-top {
  height: 40px;
  width: 40px;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip.e-tip-left {
  height: 12px;
  width: 20px;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip.e-tip-right {
  height: 12px;
  width: 20px;
}
.bubbletip.e-tooltip-wrap.e-popup {
  border: 5px solid #dfccad;
  background-color: #7b5e32;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
  height: 10px;
  width: 10px;
  border: 1px solid #dfccad;
  background-color: #7b5e32;
  border-radius: 50px;
  margin-top: 20px;
  margin-right: 20px;
}
.e-arrow-tip.e-tip-top {
  margin-left: 60px;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
  border: 1px solid #dfccad;
  border-radius: 50px;
  background-color: #7b5e32;
  width: 10px;
  height: 10px;
  margin-left: 20px;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
  border-bottom: 6px solid transparent;
  border-right: 20px solid #dfccad;
  border-top: 6px solid transparent;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
  border-bottom: 6px solid transparent;
  border-left: 20px solid #dfccad;
  border-top: 6px solid transparent;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {
  margin-top: -2px;
  margin-left: 8px;
  content: none;
  top: 1px !important;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  border: 2px solid #dfccad;
  width: 20px;
  height: 20px;
  border-radius: 50px;
  background-color: #7b5e32;
}
.bubbletip .e-arrow-tip.e-tip-top {
  left: 44px !important;
  top: -19px !important;
}
.bubbletip .e-arrow-tip.e-tip-bottom {
  top: 88.9% !important;
  left: 4px !important;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-top {
  top: 10px !important;
  border: 2px solid #dfccad;
  width: 20px;
  height: 20px;
  border-radius: 50px;
  background-color: #7b5e32;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-top:before {
  content: None;
}
.bubbletip.e-tooltip-wrap .e-tip-content {
  border-radius: inherit;
}
.bubbletip.e-tooltip-wrap.bubbletip {
  width: 150px !important;
  border-radius: 50%;
}
/* csslint ignore:end */
</style>
```

{% previewsample "page.domainurl/code-snippet/tooltip/customize-cs1" %}

Display tooltip on svg and canvas elements in Vue Tooltip component

Tooltip can be displayed on both SVG and Canvas elements. You can directly attach the <svg> or <canvas> elements to show tooltips on data visualization elements.

SVG

Create the SVG square element and refer to the following code snippet to render the tooltip on SVG square.

`ts

<ejs-tooltip target='#square' cssClass='e-tooltip-css' content='SVG Square'>

<svg>

<rect id="square" class="shape" x="50" y="20" width="50" height="50"
style="fill:#FDA600;stroke:none;stroke-width:5;stroke-opacity:0.9" />

</svg>

</ejs-tooltip>

Copyright © 2001 -2024 Syncfusion Inc.

`

Canvas

Create the canvas circle element and refer to the following code snippet to render the tooltip on Canvas circle.

`ts

<ejs-tooltip target='#circle' cssClass='e-tooltip-css' content='Canvas Circle'>

<canvas id="circle" class="shape" width="60" height="60"></canvas>

</ejs-tooltip>

`

APP.VUE

```
<template>
    <div id='app'>
        <ejs-tooltip target='#square'  cssClass='e-tooltip-css' content='SVG
Square'>
            <ejs-tooltip target='#ellipse'  cssClass='e-tooltip-css'
content='SVG Ellipse'>
                <ejs-tooltip target='#polyline'  cssClass='e-tooltip-css'
content='SVG Polyline'>
                    <ejs-tooltip target='#circle'  cssClass='e-tooltip-css'
content='Canvas Circle'>
                        <ejs-tooltip target='#triangle'  cssClass='e-
tooltip-css' content='Canvas Triangle'>
                            <div id="box" class="e-prevent-select">
                                <div class="circletool" id="rectShape"
style="left:1%;top:10%">
                                    <svg>
                                    <rect id="square" class="shape"
x="50" y="20" width="50" height="50" style="fill:#FDA600;stroke:none;stroke-
width:5;stroke-opacity:0.9" />
                                    </svg>
                                </div>
                                <div class="circletool" id="ellipseShape"
style="top:65%;">
                                    <svg>
                                        <ellipse id="ellipse" class="shape"
cx="100" cy="50" rx="20" ry="40" style="fill:#0450C2;stroke:none;stroke-
width:2" />
                                    </svg>
                                </div>
                                <div class="circletool" id="polyShape"
style="top:25%;left:40%">
                                    <svg>
                                        <polyline id="polyline"
class="shape" points="0,40 40,40 40,80 80,80 80,120 120,120 120,160"
style="fill:#ffffff;stroke:#0450C2;stroke-width:4" />
                                    </svg>
                                </div>
                                <div class="circletool" id="circleShape"
style="top:16%;left:72%">
                                    <canvas id="circle" ref="circle"
class="shape" width="60" height="60"></canvas>
```

```
                                    </div>
                                    <div class="circletool" id="triShape"
style="top:73%;left:76%">
                                        <canvas id="triangle" ref="triangle"
class="shape" width="100" height="50"></canvas>
                                    </div>
                                </div>
                            </ejs-tooltip>
                        </ejs-tooltip>
                    </ejs-tooltip>
                </ejs-tooltip>
            </ejs-tooltip>
        </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
    @media (max-width: 500px) {
    #rectShape {
        /* csslint ignore:start */
        left: -11% !important;
        /* csslint ignore:end */
    }
    #ellipseShape {
        /* csslint ignore:start */
        left: -20% !important;
        /* csslint ignore:end */
    }
    #polyShape {
        /* csslint ignore:start */
        left: 28% !important;
        /* csslint ignore:end */
    }
    #circleShape {
        /* csslint ignore:start */
        left: 68% !important;
        /* csslint ignore:end */
    }
    #triShape {
        /* csslint ignore:start */
        left: 65% !important;
        /* csslint ignore:end */
    }
    }
    @media (min-width: 500px) and (max-width: 600px) {
    #triShape {
        /* csslint ignore:start */
        left: 70% !important;
        /* csslint ignore:end */
    }
    }
    #container {
    width: 80%;
    margin: 0 auto;
    }
    .e-tooltip-css {
    filter: drop-shadow(2px 5px 5px rgba(0, 0, 0, 0.25));
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    }
    #control-container {
    /* csslint ignore:start */
    padding: 0 !important;
    /* csslint ignore:end */
    }
    #box {
    border: 1px solid #dddddd;
    background: #ffffff;
    box-sizing: border-box;
    height: 330px;
    position: relative;
    }
    .circletool {
    position: absolute;
    }
</style>
<script>
import Vue from 'vue';
import { TooltipPlugin } from '@syncfusion/ej2-vue-popups';
Vue.use(TooltipPlugin);
export default {
  data: function() {
    return {
    };
  },
  mounted: function (args) {
      let canvas = this.$refs.triangle;
      let context;
      if (canvas.getContext) {
          context = canvas.getContext('2d');
          context.beginPath();
          context.moveTo(0, 50);
          context.lineTo(100, 50);
          context.lineTo(50, 0);
          context.fillStyle = "#FDA600";
          context.fill();
      }
      canvas = this.$refs.circle;
      context = canvas.getContext('2d');
      let centerX = canvas.width / 2;
      let centerY = canvas.height / 2;
      let radius = 30;
      context.beginPath();
      context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);
      context.fillStyle = '#0450C2';
      context.fill();
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/canvas-cs1" %}

Dynamic tooltip content in Vue Tooltip component

The tooltip content can be changed dynamically using the Fetch request.

Copyright © 2001 -2024 Syncfusion Inc.

The Fetch request should be made within the beforeRender event of the tooltip. On every success, the corresponding retrieved data will be set to the content property of the tooltip.

When you hover over the icons, its respective data will be retrieved dynamically and then assigned to the tooltip's content.

Refer to the following code snippet to change the tooltip content dynamically.

`ts

onBeforeRender(args) {

this.$refs.box.content = 'Loading...';

this.$refs.box.dataBind();

let fetchApi = new Fetch('./tooltip.json', 'GET');

fetchApi.send().then(

(result) => {

for (let i = 0; i < result.length; i++) {

if (result[i].Id == args.target.id) {

/ *tslint:disable* /

this.$refs.box.content = result[i].Name;

/ *tslint:enable* /

}

}

this.$refs.box.dataBind();

},

(reason) => {

this.$refs.box.content = reason.message;

this.$refs.box.dataBind();

});

}

`

APP.VUE

```
<template>
   <div id='app'>
      <h2> Dynamic Tooltip content </h2>
      <ejs-tooltip id="box" ref='box' class="e-prevent-select"
content='Loading...' target='.circletool' showTipPointer=false v-
on:beforeRender='onBeforeRender'>
            <div id="1" class="circletool bold-01" style="display:inline-
block"  ></div>
            <div id="2" class="circletool italic" style="display:inline-
block" ></div>
```

```
            <div id="3" class="circletool underline-02"
style="display:inline-block" ></div>
            <div id="4" class="circletool cut-02" style="display:inline-
block" ></div>
            <div id="5" class="circletool copy" style="display:inline-
block"></div>
            <div id="6" class="circletool paste" style="display:inline-
block"></div>
        </ejs-tooltip>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
  h2 {
    text-align: center;
    margin-bottom: 50px;
  }
  #box {
    height: 80px;
    position: relative;
    text-align: center;
  }
  .circletool {
    height: 35px;
    width: 35px;
  }
    .underline-02 {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3c?xml
version='1.0' encoding='utf-8'?%3e%3csvg xmlns='http://www.w3.org/2000/svg'
height='32' width='32' viewBox='0 0 32 32'%3e%3cg%3e%3crect height='32'
width='32' fill='%23FFFFFF' /%3e%3cpath id='path1'
transform='rotate(0,16,16) translate(9.4685001373291,8) scale(0.5,0.5) '
fill='%23000000' d='M3.4180251,28.362L22.712003,28.362 22.712003,32
3.4180251,32z M16.780006,0L26.126,0 26.126,1.6599731
24.738978,1.6599731C23.977016,1.6599731 23.436978,1.9559937
23.024991,2.5900269 22.936002,2.723999 22.738004,3.1849976
22.738004,4.8179932L22.738004,14.367981C22.738004,16.776001
22.495025,18.684021 22.011993,20.041992 21.501983,21.440979
20.50998,22.655029 19.051974,23.64502 17.605992,24.632996
15.618996,25.127014 13.168987,25.127014 10.511032,25.127014
8.4430419,24.646973 7.0330101,23.703003 5.6169964,22.76001
4.5989924,21.466003 4.0070134,19.862976 3.6140085,18.765991
3.4139965,16.812988
3.4139968,13.888977L3.4139968,4.6790161C3.4139965,2.9509888
3.1200517,2.3649902 2.950008,2.164978 2.6440392,1.8289795
2.1230438,1.6589966 1.3900127,1.6589966L0,1.6589966 0,0.0009765625
11.186996,0.0009765625 11.186996,1.6589966
9.7790385,1.6589966C8.9750233,1.6589966 8.4330321,1.8850098
8.1130257,2.3590088 7.9830209,2.5510254 7.7700082,3.0949707
7.7700082,4.6790161L7.7700082,14.948975C7.7700082,15.838989
7.8470345,16.872009 8.0129891,18.026001 8.1730233,19.130005
8.4490233,19.984985 8.8450194,20.572021 9.2380238,21.161987
9.8149882,21.653015 10.558029,22.039978 11.313033,22.426025
12.25401,22.622009 13.381023,22.622009 14.827982,22.622009
16.134009,22.304993 17.26798,21.677979 18.374974,21.065979
19.141025,20.280029 19.545016,19.346008 19.957978,18.364014 20.16501,16.638
```

```
20.16501,14.218018L20.16501,4.6779785C20.16501,2.8049927 19.894992,2.3480225
19.808017,2.2509766 19.459994,1.8510132 18.920016,1.6589966
18.171971,1.6589966L16.780006,1.6589966 16.780006,0.0009765625z'
/%3e%3c/g%3e%3c/svg%3e ");
  }

  .bold-01 {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3c?xml
version='1.0' encoding='utf-8'?%3e%3csvg xmlns='http://www.w3.org/2000/svg'
height='32' width='32' viewBox='0 0 32 32'%3e%3cg%3e%3crect height='32'
width='32' fill='%23FFFFFF' /%3e%3cpath id='path1'
transform='rotate(0,16,16) translate(8.00000047683716,8) scale(0.5,0.5) '
fill='%23000000' d='M14.847002,16.674988C13.794,16.674988
12.893998,16.690002 12.148999,16.719971 11.572,16.75 11.086,16.779968
10.686998,16.839966L10.686998,29.120972C12.543,29.495972 14.391001,29.690979
16.178998,29.690979 19.166,29.690979 21.461002,29.075989 23.007003,27.875977
24.536002,26.70697 25.311,25.221985 25.311,23.482971 25.311,22.328003
24.955001,21.203003 24.260001,20.138977 23.570003,19.088989 22.422,18.234009
20.847003,17.619995 19.247001,17.005005 17.230001,16.674988
14.847002,16.674988z
M14.925001,2.2189941C13.375001,2.2189941,11.948999,2.3689575,10.686998,2.653
9917L10.686998,14.140015C11.231,14.230957 11.836,14.289978
12.502999,14.334961 13.309999,14.394958 14.191,14.411011 15.154001,14.411011
17.586,14.411011 19.428001,14.185974 20.629001,13.705994 21.803001,13.255981
22.715003,12.550964 23.340003,11.620972 23.969001,10.677002
24.285003,9.6270142 24.285003,8.5169678 24.285003,6.7630005 23.513,5.322998
21.928001,4.0789795 20.350001,2.848999 17.991998,2.2189941
14.925001,2.2189941z M0,0L15.384001,0C18.208003,0 20.503001,0.19500732
22.209003,0.57000732 24.834003,1.125 26.874,2.1589966 28.273002,3.6140137
29.675002,5.0830078 30.387,6.7929688 30.387,8.6969604 30.387,10.33197
29.826003,11.815979 28.719001,13.120972 27.827003,14.170959
26.578002,14.994995 24.998,15.609985 26.888,16.089966 28.363998,16.765015
29.395003,17.648987 31.123,19.118958 31.999999,20.947998 31.999999,23.092957
31.999999,24.69696 31.426002,26.271973 30.295005,27.740967
29.161002,29.226013 27.591003,30.320984 25.622001,30.994995
23.683002,31.654968 20.702003,32 16.771001,32L0,32 0,30.469971
1.6809998,30.469971C3.0099984,30.469971 3.932999,30.110962
4.4949995,29.360962 4.7329985,29.045959 5.0229994,28.205994
5.0229994,26.106995L5.0229994,5.8779907C5.0229994,3.5839844
4.6679999,2.7290039 4.370999,2.4289551 3.776997,1.8289795
2.8939973,1.5289917 1.6809998,1.5289917L0,1.5289917z' /%3e%3c/g%3e%3c/svg%3e
");
  }

  .italic {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3c?xml
version='1.0' encoding='utf-8'?%3e%3csvg xmlns='http://www.w3.org/2000/svg'
height='32' width='32' viewBox='0 0 32 32'%3e%3cg%3e%3crect height='32'
width='32' fill='%23FFFFFF' /%3e%3cpath id='path1'
transform='rotate(0,16,16) translate(9,8) scale(0.5,0.5) ' fill='%23000000'
d='M8,0L28,0 28,4 18,4 13.200012,28 20,28 20,32 0,32 0,28 9.4000244,28
14.400024,4 8,4z' /%3e%3c/g%3e%3c/svg%3e ");
  }

  .cut-02 {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3c?xml
version='1.0' encoding='utf-8'?%3e%3csvg xmlns='http://www.w3.org/2000/svg'
```

```
height='32' width='32' viewBox='0 0 32 32'%3e%3cg%3e%3crect height='32'
width='32' fill='%23FFFFFF' /%3e%3cpath id='path1'
transform='rotate(0,16,16) translate(10.9177500107586,8)
scale(0.499999940395362,0.499999940395362) ' fill='%23000000'
d='M4.2434211,24.486929C3.1234166,24.487921 1.9864136,25.411931
1.5303855,26.803925 1.2234067,27.730923 1.2683899,28.699915
1.6553864,29.461907 1.9403927,30.015906 2.3644075,30.402913
2.8894059,30.568913 4.1804323,31.000903 5.6704354,30.001914
6.2174368,28.350916 6.756443,26.700928 6.1504502,25.008931
4.8604307,24.58393 4.6584334,24.51792 4.4514308,24.486929
4.2434211,24.486929z M16.081534,24.483923C15.874531,24.484931
15.66753,24.516928 15.465532,24.58393 14.173498,25.008931
13.569489,26.700928 14.107519,28.350916 14.655497,30.001914
16.144522,30.993914 17.436526,30.568913 17.961554,30.402913
18.385539,30.015906 18.670546,29.461907 19.057541,28.699915
19.099534,27.730923 18.795546,26.803925 18.338541,25.411931
17.202546,24.481925 16.081534,24.483923z M3.8974099,0L10.162478,12.849954
16.428523,0C16.830533,0.04499804 17.086548,0.76598901 15.822529,5.3079834
14.713511,9.3009687 12.318502,14.999957 11.838487,16.182951
13.408507,19.674946 12.730492,19.907948 14.43052,21.300934
15.510515,22.189938 17.086548,23.035929 17.874548,23.649926
18.846541,24.189932 19.656546,25.15392 20.062556,26.394928
20.481566,27.649914 20.40057,28.986916 19.860558,30.066916
19.422566,30.927905 18.706557,31.561905 17.854529,31.839905
15.865529,32.497908 13.771517,31.059908 12.844507,28.767909
12.253499,27.298918 12.61248,23.269938 12.232502,21.840941
11.977496,20.886933 11.025491,19.689946 10.162478,18.668938
9.2984566,19.689946 8.3464532,20.886933 8.0924525,21.840941
7.7134518,23.262934 8.0704493,27.298918 7.4794411,28.767909
6.5534382,31.059908 4.4594264,32.488905 2.4743941,31.839905
1.6173919,31.555909 0.90339658,30.927905 0.46540476,30.066916 -
0.075615607,28.986916 -0.14760718,27.649914 0.26337645,26.386918
0.66837875,25.15392 1.4873858,24.189932 2.4573953,23.649926
3.2464034,23.035929 4.8154168,22.189938 5.8954434,21.300934
7.5944628,19.899937 6.9164481,19.674946 8.5144534,16.236952
8.0044703,14.999957 5.6114445,9.3009687 4.501419,5.3079834
3.2384078,0.76598901 3.495429,0.04499804 3.8974099,0z'
/%3e%3c/g%3e%3c/svg%3e ");
  }

  .copy {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3c?xml
version='1.0' encoding='utf-8'?%3e%3csvg xmlns='http://www.w3.org/2000/svg'
height='32' width='32' viewBox='0 0 32 32'%3e%3cg%3e%3crect height='32'
width='32' fill='%23FFFFFF' /%3e%3cpath id='path1'
transform='rotate(0,16,16) translate(9.6852502822876,8) scale(0.5,0.5) '
fill='%23000000'
d='M15.844,11.559021L15.844,15.695984C15.844,15.695984,15.764998,18.413025,1
2.761999,18.177002L9.253997,18.194031
9.253997,29.643982C9.253997,29.908997,9.4619972,30.117004,9.7219984,30.11700
4L22.904995,30.117004C23.163998,30.117004,23.371998,29.90802,23.371998,29.64
3982L23.374996,29.643982
23.374996,12.026978C23.374996,11.768982,23.166996,11.559021,22.906994,11.559
021z
M10.267996,2.28302L10.267996,7.3029785C10.267996,7.3029785,10.176001,10.5989
99,6.5360023,10.315002L2.2779993,10.333008
2.2779993,24.223022C2.2779995,24.536987,2.5360028,24.79303,2.8560022,24.7930
```

```
3L7.368003,24.79303 7.368003,17.700012 13.235998,11.47998 13.251997,11.47998
14.971998,9.6740112 19.405996,9.6740112
19.405996,2.8499756C19.405996,2.5369873,19.151999,2.28302,18.835997,2.28302z
M9.2209999,0L18.837996,0C20.415999,0,21.695997,1.2789917,21.695997,2.8530273
L21.695997,9.677002 22.902996,9.677002
22.905995,9.6749878C24.202999,9.6749878,25.258999,10.728027,25.258999,12.028
015L25.258999,29.643982C25.258999,30.945984,24.202999,32,22.905995,32L9.7229
978,32C8.4230033,32,7.3690025,30.944031,7.3690025,29.643982L7.3690025,27.078
003 2.8570017,27.078003C1.2770001,27.078003,-1.3611906E-
07,25.798035,1.4210855E-14,24.221985L1.4210855E-14,9.7319946
7.1129981,2.1940308 7.140998,2.1940308z' /%3e%3c/g%3e%3c/svg%3e ");
  }

  .paste {
    background-image: url("data:image/svg+xml;charset=UTF-8,%3c?xml
version='1.0' encoding='utf-8'?%3e%3csvg xmlns='http://www.w3.org/2000/svg'
height='32' width='32' viewBox='0 0 32 32'%3e%3cg%3e%3crect height='32'
width='32' fill='%23FFFFFF' /%3e%3cpath id='path1'
transform='rotate(0,16,16) translate(9,8) scale(0.5,0.5) ' fill='%23000000'
d='M14,18L14,30 26,30 26,22 22,22 22,18z M12,16L24,16 24,18 26,18 26,20
28,20 28,32 12,32z M0,2L2,2 2,6 20,6 20,2 22,2 22,14 10,14 10,30 0,30z
M8,0L14,0 14,2 16,2 16,4 6,4 6,2 8,2z' /%3e%3c/g%3e%3c/svg%3e ");
  }
</style>
<script>
import Vue from 'vue';
import { Fetch } from '@syncfusion/ej2-base';
import { TooltipPlugin } from '@syncfusion/ej2-vue-popups';
Vue.use(TooltipPlugin);
export default {
  data: function() {
    return {
    };
  },
  methods: {
        onBeforeRender(args) {
            this.$refs.box.content = 'Loading...';
            this.$refs.box.dataBind();
            let fetchApi = new Fetch('./tooltip.json', 'GET');
            fetchApi.send().then(
                (result) => {
                    for (let i = 0; i < result.length; i++) {
                        if (result[i].Id == args.target.id) {
                            /* tslint:disable */
                            this.$refs.box.content = result[i].Name;
                            /* tslint:enable */
                        }
                    }
                    this.$refs.box.dataBind();
                },
                (reason) => {
                    this.$refs.box.content = reason.message;
                    this.$refs.box.dataBind();
                });
        }
  }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/dynamic-content-cs1" %}

Create and show tooltip on multiple targets in Vue Tooltip component

Tooltip can be created and shown on multiple targets within a container by defining the specific target elements to the target property. So, the tooltip is initialized only on matched targets within a container.

In this case, the tooltip content is assigned from the title attribute of the target element.

`ts

<ejs-tooltip target='.e-info' position='RightCenter'>

</ejs-tooltip>

`

APP.VUE

```
<template>
   <div id='app'>
        <ejs-tooltip target='.e-info' position='RightCenter'>
            <div id="container">
                <form id="details" role="form">
                    <div id="user">
                    <div class="info">User Name:</div>
                    <div class="inputs">
                        <input type="text" ref="uname" id="uname" class="e-
info e-input" name="firstname" title="Please enter your name">
                    </div>
                    </div>
                    <br/>
                    <div>
                    <div class="info">Email Address:</div>
                    <div class="inputs">
                        <input type="text" id="mail" class="e-info e-input"
name="email" title="Enter a valid email address">
                    </div>
                    </div>
                    <br/>
                    <div>
                    <div class="info">Password:</div>
                        <div class="inputs">
                            <input id="pwd" ref="pwd" type="password"
class="e-info e-input" name="password" title="Be at least 8 characters
length">
                        </div>
                    </div>
                    <br/>
                    <div>
                    <div class="info">Confirm Password:</div>
                    <div class="inputs">
                        <input id="cpwd" ref="cpwd" type="password"
class="e-info e-input" name="Cpwd" title="Re-enter your password">
                    </div>
                    </div>
                    <br/>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                    <div class="btn">
                        <ejs-button id="sample" v-
on:click.native='onSubmit'>Submit</ejs-button>
                        <ejs-button id='clear' v-
on:click.native='onClear'type='reset'>Reset</ejs-button>
                    </div>
                </form>
            </div>
        </ejs-tooltip>
    </div>
</template>
<style>
@import "node_modules/@syncfusion/ej2-base/styles/material.css";
@import "node_modules/@syncfusion/ej2-vue-popups/styles/material.css";
#container {
    border: 1px solid #ddd;
}
#details .info {
    font-weight: bold;
    width: 165px;
    display: inline-block;
    margin-left: 10px;
}
#details .inputs {
    display: inline-block;
}
#details .btn{
  margin-top: 10px;
  position: relative;
  left: 50%;
  transform: translateX(-50%);
  display: inline-block;
}
#details #sample{
  margin-left:10px;
}
#details #clear{
  margin-left:10px;
}
#details {
  padding-top: 30px;
  padding-bottom: 30px;
  position: relative;
  left: 50%;
  transform: translateX(-50%);
  display: inline-block;
}
</style>
<script>
import Vue from 'vue';
import { TooltipPlugin } from '@syncfusion/ej2-vue-popups';
Vue.use(TooltipPlugin);
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
Vue.use(ButtonPlugin);
export default {
  data: function() {
    return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    };
  },
  methods: {
    onSubmit:function(args){
    let name = this.$refs.uname.value;
    let pwd = this.$refs.pwd.value;
    let cpwd = this.$refs.cpwd.value;
    if(name.length > 0 & name.length < 4){
        this.$refs.uname.title = 'Required Minimum 4 Characters';
        this.$refs.uname.style.backgroundColor = 'red';
        let target = this.$refs.uname;
        tooltip1.open(target);
    } else {
        this.$refs.uname.style.backgroundColor = 'white';
    }
    if(pwd !== '' && pwd.length < 8){
        this.$refs.pwd.title = 'Required Minimum 8 Characters';
        this.$refs.pwd.style.backgroundColor = 'red';
        let pwdtarget = this.$refs.pwd;
        tooltip3.open(pwdtarget);
    } else {
        this.$refs.pwd.style.backgroundColor = 'white';
    }
    if(name.length >= 4 && pwd !== '' && pwd.length >= 8  &&  pwd == cpwd ){
        alert('Form Submitted');
    } else {
        alert('Details are not Valid');
    }
    },
    onClear:function(args){
        this.$refs.uname.style.backgroundColor = 'white';
        this.$refs.pwd.style.backgroundColor = 'white';
        let target = this.$refs.uname;
        tooltip1.close(target);
        this.$refs.uname.title = 'Please enter your name';
        let pwdtarget = this.$refs.pwd;
        tooltip3.close(pwdtarget);
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/tooltip/target-cs2" %}

TreeGrid

Getting Started with the Vue Treegrid Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLILink to the Video and integrating the Syncfusion Vue Treegrid component

To get start quickly with Vue tree grid, you can check on this video:

Prerequisites

System requirements for Syncfusion Vue UI components

Copyright © 2001 -2024 Syncfusion Inc.

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

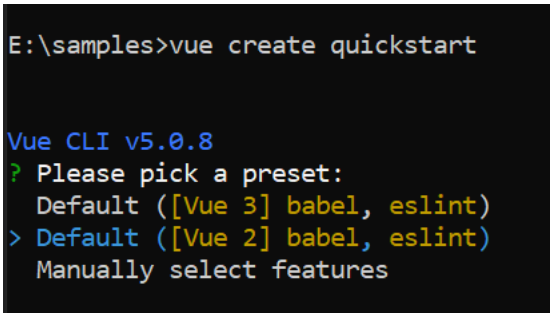
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option `Default ([Vue 2] babel, eslint)` from the menu.



Once the `quickstart` project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Treegrid component as an example. Install the `@syncfusion/ej2-vue-treegrid` package by running the following command:

`bash

npm install @syncfusion/ej2-vue-treegrid --save

`

or

`bash

yarn add @syncfusion/ej2-vue-treegrid

`

The **--save** will instruct NPM to include the TreeGrid package inside of the dependencies section of the package.json.

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

The TreeGrid has different themes. They are:

- Material
- Fabric
- Bootstrap
- High Contrast

import components CSS as given below in <style> section of the App.vue file.

~/SRC/APP.VUE

```
<style>
<!-- Material theme used for this sample -->
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-treegrid/styles/material.css";
</style>
```

Add Syncfusion Vue component

Follow the below steps to add the Vue Treegrid component:

1\. First, import and register the Treegrid component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { TreeGridComponent } from "@syncfusion/ej2-vue-treegrid";
export default {
components: {
'ejs-treegrid': TreeGridComponent
}
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

2\. In the template section, define the Treegrid component.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-treegrid></ejs-treegrid>
</div>
</template>
```

Defining Row Data

Data for the TreeGrid component is bind by using the dataSource property and value is defined in the Vue component.

It accepts either an array of JavaScript object or DataManager instance.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-treegrid :dataSource="data" childMapping="subtasks"
:treeColumnIndex="1"> </ejs-treegrid>
</div>
</template>
<script>
import { TreeGridComponent } from "@syncfusion/ej2-vue-treegrid";
let dataSource = [
{
taskID: 1,
taskName: 'Planning',
startDate: new Date('02/03/2017'),
endDate: new Date('02/07/2017'),
progress: 100,
duration: 5,
priority: 'Normal',
approved: false,
subtasks: [
{ taskID: 2, taskName: 'Plan timeline', startDate: new Date('02/03/2017'),
endDate: new Date('02/07/2017'), duration: 5, progress: 100, priority:
'Normal', approved: false },
{ taskID: 3, taskName: 'Plan budget', startDate: new Date('02/03/2017'),
endDate: new Date('02/07/2017'), duration: 5, progress: 100, approved: true
},
{ taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Critical', approved: false },
{ taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'), endDate: new Date('02/07/2017'), duration: 0, progress:
0, priority: 'Low', approved: true }
]
},
{
taskID: 6,
taskName: 'Design',
startDate: new Date('02/10/2017'),
endDate: new Date('02/14/2017'),
duration: 3,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
progress: 86,
priority: 'High',
approved: false,
subtasks: [
{ taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
60, priority: 'Normal', approved: false },
{ taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
100, priority: 'Critical', approved: false },
{ taskID: 9, taskName: 'Get approval from customer', startDate: new
Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2, progress:
100, approved: true },
{ taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2, progress:
100, approved: true },
{ taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'), endDate: new Date('02/14/2017'), duration: 0, progress:
0, priority: 'Normal', approved: true }
]
}];
export default {
components: {
'ejs-treegrid': TreeGridComponent
}
data () {
return {
data: dataSource;
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-vue-treegrid/styles/material.css";
</style>
```

Defining Columns

The TreeGrid has an option to define the columns as an array. In these columns, the following properties are used to customize the columns.

- The field has been added to map with a property name in an array of JavaScript objects.
- The headerText has been added to change the title of columns.
- The textAlign has been used to change the alignment of columns. By default, columns will be left aligned. To change the columns to right align, define the textAlign to Right.
- Also, the another useful property, format has been used. Using this, you can format number and date values to standard or custom formats. Here it is defined for the conversion of date objects to formatted strings.

Tree Column is used to expand or collapse child rows is defined by using the treeColumnIndex property.

~/SRC/APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<ejs-treegrid :dataSource='data' :treeColumnIndex='1'
childMapping='subtasks'>
<e-columns>
<e-column field='taskID' headerText='Task ID' textAlign='Right'
width=70></e-column>
<e-column field='taskName' headerText='Task Name' textAlign='Left'
width=200></e-column>
<e-column field='startDate' headerText='Start Date' textAlign='Right'
format='yMd' width=90></e-column>
<e-column field='duration' headerText='Duration' textAlign='Right'
width=80></e-column>
</e-columns>
</ejs-treegrid>
```

In the above code example, we represent the hierarchical data binding in which chilMapping property denotes the hierarchy relationship; whereas in self-referencing data binding idMapping and parentIdMapping denotes the hierarchy relationship.

Module injection

To create TreeGrid with additional features, inject the required modules. The following modules are used to extend TreeGrid's basic functionality.

- Page- Inject this module to use paging feature.
- Sort - Inject this module to use sorting feature.
- Filter - Inject this module to use filtering feature.
- ExcelExport - Inject this module to use Excel export feature.
- PdfExport - Inject this module to use PDF export feature.

Register the required array of modules under the key treegrid in the provide section.

Additional feature modules are available here

Enable Paging

The paging feature enables users to view the TreeGrid record in a paged view. It can be enabled by setting the allowPaging property to true. Also, need to inject the Page module in the provide section as follows. If the Page module is not injected, the pager will not be rendered in the TreeGrid. The pager can be customized using the pageSettings property.

In root-level paging mode, paging is based on the root-level rows only i.e., it ignores the child rows count and it can be enabled by using the pageSettings.pageSizeMode property.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex='1'
childMapping='subtasks' :allowPaging="true" :pageSettings='pageSettings'>
          <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>
    </div>
</template>
<script>
import { TreeGridComponent, ColumnsDirective, ColumnDirective, Page } from
"@syncfusion/ej2-vue-treegrid";
let dataSource = [{
            taskID: 1,
            taskName: 'Planning',
            startDate: new Date('02/03/2017'),
            endDate: new Date('02/07/2017'),
            progress: 100,
            duration: 5,
            priority: 'Normal',
            approved: false,
            subtasks: [
                { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Normal', approved: false },
                { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, approved: true },
                { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Critical', approved: false },
                { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'), endDate: new Date('02/07/2017'), duration: 0, progress:
0, priority: 'Low', approved: true }
            ]
        },
        {
            taskID: 6,
            taskName: 'Design',
            startDate: new Date('02/10/2017'),
            endDate: new Date('02/14/2017'),
            duration: 3,
            progress: 86,
            priority: 'High',
            approved: false,
            subtasks: [
                { taskID: 7, taskName: 'Software Specification', startDate:
new Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3,
progress: 60, priority: 'Normal', approved: false },
                { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
100, priority: 'Critical', approved: false },
                { taskID: 9, taskName: 'Get approval from customer',
startDate: new Date('02/13/2017'), endDate: new Date('02/14/2017'),
duration: 2, progress: 100, approved: true },
                { taskID: 10, taskName: 'Design Documentation', startDate:
new Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2,
progress: 100, approved: true },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'), endDate: new Date('02/14/2017'), duration: 0, progress:
0, priority: 'Normal', approved: true }
            ]
        },
        {
            taskID: 12,
            taskName: 'Implementation Phase',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            approved: false,
            duration: 11,
            subtasks: [
                {
                    taskID: 13,
                    taskName: 'Phase 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'High',
                    approved: false,
                    duration: 11,
                    subtasks: [{
                            taskID: 14,
                            taskName: 'Implementation Module 1',
                            startDate: new Date('02/17/2017'),
                            endDate: new Date('02/27/2017'),
                            priority: 'Normal',
                            duration: 11,
                            approved: false,
                            subtasks: [
                                { taskID: 15, taskName: 'Development Task
1', startDate: new Date('02/17/2017'), endDate: new Date('02/19/2017'),
duration: 3, progress: '50', priority: 'High', approved: false },
                                { taskID: 16, taskName: 'Development Task
2', startDate: new Date('02/17/2017'), endDate: new Date('02/19/2017'),
duration: 3, progress: '50', priority: 'Low', approved: true },
                                { taskID: 17, taskName: 'Testing',
startDate: new Date('02/20/2017'), endDate: new Date('02/21/2017'),
duration: 2, progress: '0', priority: 'Normal', approved: true },
                                { taskID: 18, taskName: 'Bug fix',
startDate: new Date('02/24/2017'), endDate: new Date('02/25/2017'),
duration: 2, progress: '0', priority: 'Critical', approved: false },
                                { taskID: 19, taskName: 'Customer review
meeting', startDate: new Date('02/26/2017'), endDate: new
Date('02/27/2017'), duration: 2, progress: '0', priority: 'High', approved:
false },
                                { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'), endDate: new Date('02/27/2017'),
duration: 0, priority: 'Low', approved: true }
                            ]
                        }]
                }]
        }];
export default {
  components: {
    'ejs-treegrid': TreeGridComponent,
```

```
    'e-column': ColumnDirective,
    'e-columns': ColumnsDirective
  },
  data() {
    return {
      data: dataSource,
      pageSettings: { pageSize: 7 }
    };
  },
  provide: {
    treegrid: [Page]
  }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-treegrid/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/getting-started/default-cs1" %}

Enable Sorting

The sorting feature enables the user to order the records. It can be enabled by setting allowSorting property to true. Also, need to inject the Sort module in the provide section as follow.

If we didn't inject the Sort module, then user not able to sort when click on headers. Sorting feature can be customized using sortSettings property.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex='1'
:allowSorting='true'
          :sortSettings='sortSettings'
          childMapping='subtasks' :allowPaging="true"
:pageSettings='pageSettings'>
          <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
</template>
<script>
import { TreeGridComponent, ColumnDirective, ColumnsDirective, Page, Sort }
from "@syncfusion/ej2-vue-treegrid";
let dataSource = [{
            taskID: 1,
            taskName: 'Planning',
            startDate: new Date('02/03/2017'),
            endDate: new Date('02/07/2017'),
            progress: 100,
            duration: 5,
            priority: 'Normal',
            approved: false,
            subtasks: [
                { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Normal', approved: false },
                { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, approved: true },
                { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Critical', approved: false },
                { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'), endDate: new Date('02/07/2017'), duration: 0, progress:
0, priority: 'Low', approved: true }
            ]
        },
        {
            taskID: 6,
            taskName: 'Design',
            startDate: new Date('02/10/2017'),
            endDate: new Date('02/14/2017'),
            duration: 3,
            progress: 86,
            priority: 'High',
            approved: false,
            subtasks: [
                { taskID: 7, taskName: 'Software Specification', startDate:
new Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3,
progress: 60, priority: 'Normal', approved: false },
                { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
100, priority: 'Critical', approved: false },
                { taskID: 9, taskName: 'Get approval from customer',
startDate: new Date('02/13/2017'), endDate: new Date('02/14/2017'),
duration: 2, progress: 100, approved: true },
                { taskID: 10, taskName: 'Design Documentation', startDate:
new Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2,
progress: 100, approved: true },
                { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'), endDate: new Date('02/14/2017'), duration: 0, progress:
0, priority: 'Normal', approved: true }
            ]
        },
        {
```

```
            taskID: 12,
            taskName: 'Implementation Phase',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            approved: false,
            duration: 11,
            subtasks: [
                {
                    taskID: 13,
                    taskName: 'Phase 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'High',
                    approved: false,
                    duration: 11,
                    subtasks: [{
                            taskID: 14,
                            taskName: 'Implementation Module 1',
                            startDate: new Date('02/17/2017'),
                            endDate: new Date('02/27/2017'),
                            priority: 'Normal',
                            duration: 11,
                            approved: false,
                            subtasks: [
                                { taskID: 15, taskName: 'Development Task
1', startDate: new Date('02/17/2017'), endDate: new Date('02/19/2017'),
duration: 3, progress: '50', priority: 'High', approved: false },
                                { taskID: 16, taskName: 'Development Task
2', startDate: new Date('02/17/2017'), endDate: new Date('02/19/2017'),
duration: 3, progress: '50', priority: 'Low', approved: true },
                                { taskID: 17, taskName: 'Testing',
startDate: new Date('02/20/2017'), endDate: new Date('02/21/2017'),
duration: 2, progress: '0', priority: 'Normal', approved: true },
                                { taskID: 18, taskName: 'Bug fix',
startDate: new Date('02/24/2017'), endDate: new Date('02/25/2017'),
duration: 2, progress: '0', priority: 'Critical', approved: false },
                                { taskID: 19, taskName: 'Customer review
meeting', startDate: new Date('02/26/2017'), endDate: new
Date('02/27/2017'), duration: 2, progress: '0', priority: 'High', approved:
false },
                                { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'), endDate: new Date('02/27/2017'),
duration: 0, priority: 'Low', approved: true }
                            ]
                        }]
                }]
        }];
export default {
  components: {
    'ejs-treegrid': TreeGridComponent,
    'e-column': ColumnDirective,
    'e-columns': ColumnsDirective
  },
  data() {
    return {
      data: dataSource,
```

```
      sortSettings: { columns: [{ field: 'taskName', direction: 'Ascending'
}, { field: 'taskID', direction: 'Descending' }]  },
      pageSettings: { pageSize: 7 }
    };
  },
  provide: {
    treegrid: [Page, Sort]
  }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-treegrid/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/getting-started/default-cs2" %}

Enable Filtering

The filtering feature enables you to view the reduced amount of records based on the filter criteria. It can be enabled by setting the allowFiltering property to true. Also, need to inject the Filter module in the provide section as follow. If Filter module is not injected, filter bar will not be rendered in TreeGrid. Filtering feature can be customized using the filterSettings property.

By default, filtered records are shown along with its parent records. This behavior can be changed by using the filterSettings-hierarchyMode property.

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex='1'
:allowFiltering='true'
          :sortSettings='sortSettings'
          :allowSorting='true' childMapping='subtasks' :allowPaging="true"
:pageSettings='pageSettings'>
          <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>
    </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import { TreeGridComponent, ColumnDirective, ColumnsDirective, Page, Sort,
Filter } from "@syncfusion/ej2-vue-treegrid";
let dataSource = [{
            taskID: 1,
            taskName: 'Planning',
            startDate: new Date('02/03/2017'),
            endDate: new Date('02/07/2017'),
            progress: 100,
            duration: 5,
            priority: 'Normal',
            approved: false,
            subtasks: [
                { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Normal', approved: false },
                { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, approved: true },
                { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Critical', approved: false },
                { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'), endDate: new Date('02/07/2017'), duration: 0, progress:
0, priority: 'Low', approved: true }
            ]
        },
        {
            taskID: 6,
            taskName: 'Design',
            startDate: new Date('02/10/2017'),
            endDate: new Date('02/14/2017'),
            duration: 3,
            progress: 86,
            priority: 'High',
            approved: false,
            subtasks: [
                { taskID: 7, taskName: 'Software Specification', startDate:
new Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3,
progress: 60, priority: 'Normal', approved: false },
                { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
100, priority: 'Critical', approved: false },
                { taskID: 9, taskName: 'Get approval from customer',
startDate: new Date('02/13/2017'), endDate: new Date('02/14/2017'),
duration: 2, progress: 100, approved: true },
                { taskID: 10, taskName: 'Design Documentation', startDate:
new Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2,
progress: 100, approved: true },
                { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'), endDate: new Date('02/14/2017'), duration: 0, progress:
0, priority: 'Normal', approved: true }
            ]
        },
        {
            taskID: 12,
            taskName: 'Implementation Phase',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            approved: false,
            duration: 11,
            subtasks: [
                {
                    taskID: 13,
                    taskName: 'Phase 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'High',
                    approved: false,
                    duration: 11,
                    subtasks: [{
                            taskID: 14,
                            taskName: 'Implementation Module 1',
                            startDate: new Date('02/17/2017'),
                            endDate: new Date('02/27/2017'),
                            priority: 'Normal',
                            duration: 11,
                            approved: false,
                            subtasks: [
                                { taskID: 15, taskName: 'Development Task
1', startDate: new Date('02/17/2017'), endDate: new Date('02/19/2017'),
duration: 3, progress: '50', priority: 'High', approved: false },
                                { taskID: 16, taskName: 'Development Task
2', startDate: new Date('02/17/2017'), endDate: new Date('02/19/2017'),
duration: 3, progress: '50', priority: 'Low', approved: true },
                                { taskID: 17, taskName: 'Testing',
startDate: new Date('02/20/2017'), endDate: new Date('02/21/2017'),
duration: 2, progress: '0', priority: 'Normal', approved: true },
                                { taskID: 18, taskName: 'Bug fix',
startDate: new Date('02/24/2017'), endDate: new Date('02/25/2017'),
duration: 2, progress: '0', priority: 'Critical', approved: false },
                                { taskID: 19, taskName: 'Customer review
meeting', startDate: new Date('02/26/2017'), endDate: new
Date('02/27/2017'), duration: 2, progress: '0', priority: 'High', approved:
false },
                                { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'), endDate: new Date('02/27/2017'),
duration: 0, priority: 'Low', approved: true }
                            ]
                        }]
                }]
        }];
export default {
  components: {
    'ejs-treegrid': TreeGridComponent,
    'e-column': ColumnDirective,
    'e-columns': ColumnsDirective
  },
  data() {
    return {
      data: dataSource,
      sortSettings: { columns: [{ field: 'taskName', direction: 'Ascending'
}, { field: 'taskID', direction: 'Descending' }]  },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      pageSettings: { pageSize: 7 }
    };
  },
  provide: {
    treegrid: [Page, Sort, Filter]
  }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-treegrid/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/getting-started/default-cs3" %}

Run the application

The Vue TreeGrid application is configured to compile and run the application in a browser. Use the following command to run the application.

`bash

npm run dev

`

~/SRC/APP.VUE

```
<template>
    <div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex='1'
:allowFiltering='true' :allowSorting='true'
          :sortSettings='sortSettings'
          childMapping='subtasks' :allowPaging="true"
:pageSettings='pageSettings'>
          <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { TreeGridComponent, ColumnDirective, ColumnsDirective, Page, Sort,
Filter } from "@syncfusion/ej2-vue-treegrid";
let dataSource = [{
            taskID: 1,
            taskName: 'Planning',
            startDate: new Date('02/03/2017'),
            endDate: new Date('02/07/2017'),
            progress: 100,
            duration: 5,
            priority: 'Normal',
            approved: false,
            subtasks: [
                { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Normal', approved: false },
                { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, approved: true },
                { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Critical', approved: false },
                { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'), endDate: new Date('02/07/2017'), duration: 0, progress:
0, priority: 'Low', approved: true }
            ]
        },
        {
            taskID: 6,
            taskName: 'Design',
            startDate: new Date('02/10/2017'),
            endDate: new Date('02/14/2017'),
            duration: 3,
            progress: 86,
            priority: 'High',
            approved: false,
            subtasks: [
                { taskID: 7, taskName: 'Software Specification', startDate:
new Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3,
progress: 60, priority: 'Normal', approved: false },
                { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
100, priority: 'Critical', approved: false },
                { taskID: 9, taskName: 'Get approval from customer',
startDate: new Date('02/13/2017'), endDate: new Date('02/14/2017'),
duration: 2, progress: 100, approved: true },
                { taskID: 10, taskName: 'Design Documentation', startDate:
new Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2,
progress: 100, approved: true },
                { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'), endDate: new Date('02/14/2017'), duration: 0, progress:
0, priority: 'Normal', approved: true }
            ]
        },
        {
            taskID: 12,
            taskName: 'Implementation Phase',
            startDate: new Date('02/17/2017'),
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            approved: false,
            duration: 11,
            subtasks: [
                {
                    taskID: 13,
                    taskName: 'Phase 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'High',
                    approved: false,
                    duration: 11,
                    subtasks: [{
                            taskID: 14,
                            taskName: 'Implementation Module 1',
                            startDate: new Date('02/17/2017'),
                            endDate: new Date('02/27/2017'),
                            priority: 'Normal',
                            duration: 11,
                            approved: false,
                            subtasks: [
                                { taskID: 15, taskName: 'Development Task
1', startDate: new Date('02/17/2017'), endDate: new Date('02/19/2017'),
duration: 3, progress: '50', priority: 'High', approved: false },
                                { taskID: 16, taskName: 'Development Task
2', startDate: new Date('02/17/2017'), endDate: new Date('02/19/2017'),
duration: 3, progress: '50', priority: 'Low', approved: true },
                                { taskID: 17, taskName: 'Testing',
startDate: new Date('02/20/2017'), endDate: new Date('02/21/2017'),
duration: 2, progress: '0', priority: 'Normal', approved: true },
                                { taskID: 18, taskName: 'Bug fix',
startDate: new Date('02/24/2017'), endDate: new Date('02/25/2017'),
duration: 2, progress: '0', priority: 'Critical', approved: false },
                                { taskID: 19, taskName: 'Customer review
meeting', startDate: new Date('02/26/2017'), endDate: new
Date('02/27/2017'), duration: 2, progress: '0', priority: 'High', approved:
false },
                                { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'), endDate: new Date('02/27/2017'),
duration: 0, priority: 'Low', approved: true }
                            ]
                        }]
                }]
        }];
export default {
  components: {
    'ejs-treegrid': TreeGridComponent,
    'e-column': ColumnDirective,
    'e-columns': ColumnsDirective
  },
  data() {
    return {
      data: dataSource,
      sortSettings: { columns: [{ field: 'taskName', direction: 'Ascending'
}, { field: 'taskID', direction: 'Descending' }]  },
      pageSettings: { pageSize: 7 }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    };
  },
  provide: {
    treegrid: [Page, Sort, Filter]
  }
}
</script>
<style>
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import "../node_modules/@syncfusion/ej2-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-treegrid/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/getting-started/default-cs4" %}

Getting Started with the Vue Tree Grid Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Tree Grid component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

Copyright © 2001 -2024 Syncfusion Inc.

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

Copyright © 2001 -2024 Syncfusion Inc.

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Grid component as an example. To use the Vue Tree Grid component in the project, the @syncfusion/ej2-vue-treegrid package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-treegrid --save

`

or

`bash

yarn add @syncfusion/ej2-vue-treegrid

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Tree Grid component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-treegrid/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue Tree Grid component using Composition API or Options API:

Copyright © 2001 -2024 Syncfusion Inc.

1.First, import and register the Tree Grid component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { TreeGridComponent as EjsTreegrid, ColumnsDirective as EColumns,
ColumnDirective as EColumn } from '@syncfusion/ej2-vue-treegrid';
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { TreeGridComponent, ColumnsDirective, ColumnDirective } from
'@syncfusion/ej2-vue-treegrid';
//Component registration
export default {
name: "App",
components: {
'ejs-treegrid' : TreeGridComponent,
"e-columns": ColumnsDirective,
"e-column": ColumnDirective
}
}
</script>
```

2.In the template section, define the Tree Grid component with the dataSource property and column definitions.

~/SRC/APP.VUE

```
<template>
<ejs-treegrid :dataSource='data' :treeColumnIndex='1'
childMapping='subtasks'>
<e-columns>
<e-column field='taskID' headerText='Task ID' textAlign='Right'
width=70></e-column>
<e-column field='taskName' headerText='Task Name' textAlign='Left'
width=200></e-column>
<e-column field='startDate' headerText='Start Date' textAlign='Right'
format='yMd' width=90></e-column>
<e-column field='duration' headerText='Duration' textAlign='Right'
width=80></e-column>
</e-columns>
</ejs-treegrid>
</template>
```

3.Declare the values for the dataSource property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const data = [
{
```

Copyright © 2001 -2024 Syncfusion Inc.

```
taskID: 1,
taskName: 'Planning', startDate: new Date('02/03/2017'), endDate: new
Date('02/07/2017'),      progress: 100, duration: 5, priority: 'Normal',
approved: false,
subtasks: [
{ taskID: 2, taskName: 'Plan timeline', startDate: new Date('02/03/2017'),
endDate: new Date('02/07/2017'), duration: 5, progress: 100, priority:
'Normal', approved: false },
{ taskID: 3, taskName: 'Plan budget', startDate: new Date('02/03/2017'),
endDate: new Date('02/07/2017'), duration: 5, progress: 100, approved: true
},
{ taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Critical', approved: false },
{ taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'), endDate: new Date('02/07/2017'), duration: 0, progress:
0, priority: 'Low', approved: true }
]
},
{
taskID: 6, taskName: 'Design', startDate: new Date('02/10/2017'), endDate:
new Date('02/14/2017'), duration: 3, progress: 86, priority: 'High',
approved: false,
subtasks: [
{ taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
60, priority: 'Normal', approved: false },
{ taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
100, priority: 'Critical', approved: false },
{ taskID: 9, taskName: 'Get approval from customer', startDate: new
Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2, progress:
100, approved: true },
{ taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2, progress:
100, approved: true },
{ taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'), endDate: new Date('02/14/2017'), duration: 0, progress:
0, priority: 'Normal', approved: true }
]
}
];
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
data:[
{
taskID: 1,
taskName: 'Planning', startDate: new Date('02/03/2017'), endDate: new
Date('02/07/2017'),      progress: 100, duration: 5, priority: 'Normal',
approved: false,
subtasks: [
```

```
{ taskID: 2, taskName: 'Plan timeline', startDate: new Date('02/03/2017'),
endDate: new Date('02/07/2017'), duration: 5, progress: 100, priority:
'Normal', approved: false },
{ taskID: 3, taskName: 'Plan budget', startDate: new Date('02/03/2017'),
endDate: new Date('02/07/2017'), duration: 5, progress: 100, approved: true
},
{ taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Critical', approved: false },
{ taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'), endDate: new Date('02/07/2017'), duration: 0, progress:
0, priority: 'Low', approved: true }
]
},
{
taskID: 6, taskName: 'Design', startDate: new Date('02/10/2017'), endDate:
new Date('02/14/2017'), duration: 3, progress: 86, priority: 'High',
approved: false,
subtasks: [
{ taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
60, priority: 'Normal', approved: false },
{ taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
100, priority: 'Critical', approved: false },
{ taskID: 9, taskName: 'Get approval from customer', startDate: new
Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2, progress:
100, approved: true },
{ taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2, progress:
100, approved: true },
{ taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'), endDate: new Date('02/14/2017'), duration: 0, progress:
0, priority: 'Normal', approved: true }
]
}
],
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-treegrid :dataSource='data' :treeColumnIndex='1'
childMapping='subtasks'>
<e-columns>
<e-column field='taskID' headerText='Task ID' textAlign='Right'
width=70></e-column>
<e-column field='taskName' headerText='Task Name' textAlign='Left'
width=200></e-column>
<e-column field='startDate' headerText='Start Date' textAlign='Right'
format='yMd' width=90></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-column field='duration' headerText='Duration' textAlign='Right'
width=80></e-column>
</e-columns>
</ejs-treegrid>
</template>
<script setup>
import { TreeGridComponent as EjsTreegrid, ColumnsDirective as EColumns,
ColumnDirective as EColumn } from '@syncfusion/ej2-vue-treegrid';
const data = [
{
taskID: 1,
taskName: 'Planning', startDate: new Date('02/03/2017'), endDate: new
Date('02/07/2017'),      progress: 100, duration: 5, priority: 'Normal',
approved: false,
subtasks: [
{ taskID: 2, taskName: 'Plan timeline', startDate: new Date('02/03/2017'),
endDate: new Date('02/07/2017'), duration: 5, progress: 100, priority:
'Normal', approved: false },
{ taskID: 3, taskName: 'Plan budget', startDate: new Date('02/03/2017'),
endDate: new Date('02/07/2017'), duration: 5, progress: 100, approved: true
},
{ taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Critical', approved: false },
{ taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'), endDate: new Date('02/07/2017'), duration: 0, progress:
0, priority: 'Low', approved: true }
]
},
{
taskID: 6, taskName: 'Design', startDate: new Date('02/10/2017'), endDate:
new Date('02/14/2017'), duration: 3, progress: 86, priority: 'High',
approved: false,
subtasks: [
{ taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
60, priority: 'Normal', approved: false },
{ taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
100, priority: 'Critical', approved: false },
{ taskID: 9, taskName: 'Get approval from customer', startDate: new
Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2, progress:
100, approved: true },
{ taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2, progress:
100, approved: true },
{ taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'), endDate: new Date('02/14/2017'), duration: 0, progress:
0, priority: 'Normal', approved: true }
]
},
];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-treegrid/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-treegrid :dataSource='data' :treeColumnIndex='1'
childMapping='subtasks'>
<e-columns>
<e-column field='taskID' headerText='Task ID' textAlign='Right'
width=70></e-column>
<e-column field='taskName' headerText='Task Name' textAlign='Left'
width=200></e-column>
<e-column field='startDate' headerText='Start Date' textAlign='Right'
format='yMd' width=90></e-column>
<e-column field='duration' headerText='Duration' textAlign='Right'
width=80></e-column>
</e-columns>
</ejs-treegrid>
</template>
<script>
import { TreeGridComponent, ColumnsDirective, ColumnDirective } from
'@syncfusion/ej2-vue-treegrid';
let dataSource = [{
taskID: 1,
taskName: 'Planning', startDate: new Date('02/03/2017'), endDate: new
Date('02/07/2017'), progress: 100, duration: 5, priority: 'Normal',
approved: false,
subtasks: [
{ taskID: 2, taskName: 'Plan timeline', startDate: new Date('02/03/2017'),
endDate: new Date('02/07/2017'), duration: 5, progress: 100, priority:
'Normal', approved: false },
{ taskID: 3, taskName: 'Plan budget', startDate: new Date('02/03/2017'),
endDate: new Date('02/07/2017'), duration: 5, progress: 100, approved: true
},
{ taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'), endDate: new Date('02/07/2017'), duration: 5, progress:
100, priority: 'Critical', approved: false },
{ taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'), endDate: new Date('02/07/2017'), duration: 0, progress:
0, priority: 'Low', approved: true }
]
},
{
taskID: 6, taskName: 'Design', startDate: new Date('02/10/2017'), endDate:
new Date('02/14/2017'), duration: 3, progress: 86, priority: 'High',
approved: false,
subtasks: [
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
60, priority: 'Normal', approved: false },
{ taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'), endDate: new Date('02/12/2017'), duration: 3, progress:
100, priority: 'Critical', approved: false },
{ taskID: 9, taskName: 'Get approval from customer', startDate: new
Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2, progress:
100, approved: true },
{ taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'), endDate: new Date('02/14/2017'), duration: 2, progress:
100, approved: true },
{ taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'), endDate: new Date('02/14/2017'), duration: 0, progress:
0, priority: 'Normal', approved: true }
]
},
];
export default {
// Component registration
name: "App",
// Declaring component and its directives
components: {
'ejs-treegrid': TreeGridComponent,
"e-columns": ColumnsDirective,
"e-column": ColumnDirective,
},
// Bound properties declaration
data() {
return {
data: dataSource
};
},
provide: {
treegrid: []
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-calendars/styles/material.css";
@import "../node_modules/@syncfusion/ej2-dropdowns/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-treegrid/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

Copyright © 2001 -2024 Syncfusion Inc.

npm run dev

`

or

`bash

yarn run dev

`

The output will appear as follows:

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Module in Vue Treegrid component

The following value providers should be injected to extend TreeGrid's functionality.

| Module | Description |

|------|------------|

| PageService| Inject this module to use paging feature.|

| SortService| Inject this module to use sorting feature.|

| FilterService| Inject this module to use filtering feature.|

| EditService| Inject this module to use editing feature.|

| AggregateService| Inject this module to use aggregate feature.|

| ColumnMenuService| Inject this module to use column menu feature.|

| CommandColumnService| Inject this module to use command column feature.|

| ContextMenuService| Inject this module to use context menu feature.

| ResizeService| Inject this module to use resize feature.|

| ReorderService| Inject this module to use reorder feature.|

Copyright © 2001 -2024 Syncfusion Inc.

| PrintService | Inject this module to use to use print feature and this is a default injected module. |

| ToolbarService | Inject this module to use toolbar feature. |

| ExcelExportService | Inject this module to use Excel export feature. |

| PdfExportService | Inject this module to use PDF export feature. |

These modules should be injected into the provide section and use treegrid as a key of the object.

Data Binding

Data binding in Vue Treegrid component

The TreeGrid uses DataManager, which supports both RESTful JSON data services binding and local JavaScript object array binding. The dataSource property can be assigned either with the instance of Link to the Video or JavaScript object array collection.

It supports two kinds of data binding method:

- Local data
- Remote data

To learn about tree grid data binding quickly, you can check on this video:

Binding with ajax

You can use TreeGrid dataSource property to bind the data source to TreeGrid from external Fetch request. In the below code we have fetched the data source from the server with the help of Fetch request and provided that to dataSource property by using onSuccess event of the Fetch.

APP.VUE

```
<template>
  <div id="app">
    <ejs-button v-on:click.native="btnClick">Bind Data</ejs-button>
        <ejs-treegrid ref=treegrid :dataSource="data" idMapping='TaskID'
parentIdMapping='ParentItem'  :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings'>
            <e-columns>
                <e-column field='TaskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='TaskName' headerText='Task Name'
width='100'></e-column>
                <e-column field='Duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
                </e-columns>
                <e-column field='Progress' headerText='Progress' width='80'
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { Fetch } from '@syncfusion/ej2-base';
Vue.use(TreeGridPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(ButtonPlugin);
export default {
  data () {
    return {
      data: {},
      pageSettings: { pageCount: 4 }
    };
  },
  provide: {
      treegrid: [ Page ]
    },
   methods:{
         btnClick: function (args){
        let treegrid = document.getElementsByClassName("e-
treegrid")[0].ej2_instances[0]; // TreeGrid instance
        let fetch = new Fetch
("https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData", "GET");treegrid.showSpinner();
        fetch.send();
        fetch.onSuccess = function (result) {
        treegrid.hideSpinner();
        treegrid.dataSource = result;
        };
      }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/data-binding/default-cs1" %}

* If you bind the dataSource from this way, then it acts like a local dataSource. So you cannot perform any server side crud actions.

Handling expandStateMapping

To denotes the expand status of parent row, define the expandStateMapping property of tree grid. The expandStateMapping property maps the field name in data source, that denotes whether parent record is in expanded or collapsed state and this is useful to renders parent row in expanded or collapsed state based on this mapping property value in data source.

`

<template>

<ejs-treegrid :dataSource='data' idMapping='TaskID' parentIdMapping='ParentValue' hasChildMapping='isParent' :treeColumnIndex='1' expandStateMapping='IsExpanded' height='400px'>

<e-columns>

<e-column field='TaskID' headerText='Task ID' width=90 textAlign='Right'></e-column>

<e-column field='TaskName' headerText='Task Name' width=180></e-column>

<e-column field='Duration' headerText='Duration' width=80></e-column>

</e-columns>

Copyright © 2001 -2024 Syncfusion Inc.

```
</ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Vue.use(TreeGridPlugin);
export default {
data() {
return {
data: new DataManager({
url: "Home/DataSource",
adaptor: new UrlAdaptor
})
};
},
}
</script>
```

The following code example defines expandStateMapping property at server end.

```ts
public ActionResult ExpandStateMapping()
{
return View();
}
public class TreeData
{
public static List<TreeData> tree = new List<TreeData>();
[System.ComponentModel.DataAnnotations.Key]
public int TaskID { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
public int? ParentValue { get; set; }
public bool? isParent { get; set; }
public bool IsExpanded { get; set; }
public TreeData() { }
public static List<TreeData> GetTree()
{
if (tree.Count == 0)
{
int root = 0;
for (var t = 1; t <= 500; t++)
{
Random ran = new Random();
string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ? "Release Breaker" : "Critical";
string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ? "Open" : "In Progress";
root++;
int rootItem = root;
tree.Add(new TreeData() { TaskID = rootItem, TaskName = "Parent task " + rootItem.ToString(), isParent
= true, IsExpanded = false, ParentValue = null, Duration = ran.Next(1, 50) });
int parent = root;
for (var d = 0; d < 1; d++)
{
root++;
string value = ((parent + 1) % 3 == 0) ? "Low" : "Critical";
int par = parent + 1;
progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";
int iD = root;
tree.Add(new TreeData() { TaskID = iD, TaskName = "Child task " + iD.ToString(), isParent = true,
IsExpanded = false, ParentValue = rootItem, Duration = ran.Next(1, 50) });
int subparent = root;
for (var c = 0; c < 500; c++)
{
root++;
string val = ((subparent + c + 1) % 3 == 0) ? "Low" : "Critical";
int subchild = subparent + c + 1;
```

string progress = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";

int childID = root ;

tree.Add(new TreeData() { TaskID = childID, TaskName = "sub Child task " + childID.ToString(), isParent = false, IsExpanded = false, ParentValue = subparent, Duration = ran.Next(1, 50) });

}

}

}

}

return tree;

}

}

`

Local data in Vue Treegrid component

In Local Data binding, data source for rendering the TreeGrid control is retrieved from the same application locally.

Two types of Data binding are possible with the TreeGrid control.

- Hierarchical Datasource binding
- Self-Referential Data binding (Flat Data)

To bind local data to the treegrid, you can assign a JavaScript object array to the dataSource property. The local data source can also be provided as an instance of the DataManager.

By default, DataManager uses JsonAdaptor for local data-binding.

Hierarchy data source binding

The childMapping property is used to map the child records in hierarchy data source.

The following code example shows you how to bind the hierarchical local data into the TreeGrid control.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' :allowPaging='true' :pageSettings='pageSettings'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=70
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=200></e-column>
                <e-column field='startDate' headerText='Start Date' width=90
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from './datasource.js';
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: sampleData,
      pageSettings: { pageSize: 7 }
    };
  },
  provide: {
      treegrid: [ Page ]
    }
}
</script>
> * Remote data binding is not supported for Hierarchy Data.
```

{% previewsample "page.domainurl/code-snippet/treegrid/data-binding/default-cs2" %}

Self-Referential data binding (Flat data)

TreeGrid is rendered from Self-Referential data structures by providing two fields, ID field and parent ID field.

- **ID Field**: This field contains unique values used to identify nodes. Its name is assigned to the idMapping property.
- **Parent ID Field**: This field contains values that indicate parent nodes. Its name is assigned to the parentIdMapping property.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" idMapping='TaskID'
parentIdMapping='parentID' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings'>
            <e-columns>
                <e-column field='TaskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='TaskName' headerText='Task Name'
width=180></e-column>
                <e-column field='StartDate' headerText='Start Date' width=90
format="yMd" textAlign='Right'></e-column>
                <e-column field='Duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
```

```
import { projectData } from './datasource.js';
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: projectData,
      pageSettings: { pageSize: 7 }
    };
  },
  provide: {
      treegrid: [ Page ]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/data-binding/default-cs3" %}

Herewith we have provided list of reserved properties and the purpose used in TreeGrid. We recommend to avoid these reserved properties for Internal purpose(To get rid of conflicts).

Reserved keywords | Purpose

childRecords | Specifies the childRecords of a parentData

hasChildRecords | Specifies whether the record contains child records

hasFilteredChildRecords | Specifies whether the record contains filtered child records

expanded | Specifies whether the child records are expanded

parentItem | Specifies the parentItem of childRecords

index | Specifies the index of current record

level | Specifies the hierarchy level of record

filterLevel | Specifies the hierarchy level of filtered record

parentIdMapping | Specifies the parentID

uniqueID | Specifies the unique ID of a record

parentUniqueID | Specifies the parent Unique ID of a record

checkboxState | Specifies the checkbox state of a record

isSummaryRow | Specifies the summary of a record

taskData | Specifies the main data

primaryParent | Specifies the Primary data

Remote data in Vue Treegrid component

To bind remote data to TreeGrid component, assign service data as an instance of DataManager to the dataSource property. To interact with remote data source, provide the endpoint url and define the hasChildMapping property of treegrid.

The hasChildMapping property maps the field name in data source, that denotes whether current record holds any child records. This is useful internally to show expand icon while binding child data on demand.

Copyright © 2001 -2024 Syncfusion Inc.

The TreeGrid provides Load on Demand support for rendering remote data. The Load on demand is considered in TreeGrid for the following actions.

- Expanding root nodes.
- Navigating pages, with paging enabled in TreeGrid.

When load on demand is enabled, all the root nodes are rendered in collapsed state at initial load.

When load on demand support is enabled in TreeGrid with paging, the current or active page's root node alone will be rendered in collapsed state. On expanding the root node, the child nodes will be loaded from the remote server.

When a root node is expanded, its child nodes are rendered and are cached locally, such that on consecutive expand/collapse actions on root node, the child nodes are loaded from the cache instead from the remote server.

Similarly, if the user navigates to a new page, the root nodes of that specific page, will be rendered with request to the remote server.

Remote Data Binding supports only Self-Referential Data and by default the pageSizeMode for Remote Data is Root mode. i.e only root node's count will be shown in pager while using Remote Data

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" idMapping='TaskID'
parentIdMapping='ParentID' :treeColumnIndex='1' :allowPaging='true'>
            <e-columns>
                <e-column field='TaskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='TaskName' headerText='Task Name'
width=180></e-column>
                <e-column field='StartDate' headerText='Start Date' width=90
format="yMd" textAlign='Right'></e-column>
                <e-column field='Duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { DataManager, WebApiAdaptor } from "@syncfusion/ej2-data";
Vue.use(TreeGridPlugin);
export default {
   data() {
    let SERVICE_URI ="https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData";
    return {
      data: new DataManager({
        url: SERVICE_URI,
        adaptor: new WebApiAdaptor(),
        crossDomain: true
      })
```

```
    };
  }
  provide: {
      treegrid: [ Page ]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/data-binding/default-cs4" %}

By default, DataManager uses ODataAdaptor for remote data-binding.

Based on the RESTful web services, set the corresponding adaptor to DataManager. Refer here for more details.

Filtering and searching server-side data operations are not supported in load on demand

LoadChildOnDemand

While binding remote data to Tree Grid component, by default Tree Grid renders parent rows in collapsed state. Tree Grid provides option to load the child records also during the initial rendering itself for remote data binding by setting loadChildOnDemand as true.

When loadChildOnDemand is enabled parent records are rendered in expanded state.

The following code example describes the behavior of the loadChildOnDemand feature of Tree Grid.

`

<template>

<ejs-treegrid :dataSource='data' idMapping='TaskID' parentIdMapping='parentID' :treeColumnIndex='1' :allowPaging='true' :loadChildOnDemand='true' height='270px'>

<e-columns>

<e-column field='TaskID' :isPrimaryKey='true' headerText='Task ID' width=90 textAlign='Right'></e-column>

<e-column field='TaskName' headerText='Task Name' width=180></e-column>

<e-column field='StartDate' headerText='Start Date' editType= 'datepickeredit' type= 'date' format= 'yMd' textAlign='Right' width=170></e-column>

<e-column field='EndDate' headerText='End Date' editType= 'datepickeredit' type= 'date' format= 'yMd' textAlign='Right' width=170></e-column>

<e-column field='Priority' headerText='Priority' width=110></e-column>

</ejs-treegrid>

</template>

<script>

import Vue from "vue";

Copyright © 2001 -2024 Syncfusion Inc.

```
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Vue.use(TreeGridPlugin);
export default {
data() {
return {
data: new DataManager({
url: "Home/DataSource",
updateUrl: "Home/Update",
insertUrl: "Home/Insert",
removeUrl: "Home/Delete",
batchUrl: "Home/Remove",
adaptor: new UrlAdaptor
})
};
},
provide: {
treegrid: [ Page ]
}
}
</script>
`
```

Also while using **loadChildOnDemand** we need to handle the child records on server end and it is applicable to CRUD operations also.

The following code example describes handling of child records at server end.

```ts
public ActionResult UrlDatasource(DataManagerRequest dm)
{
if (TreeData.tree.Count == 0)
TreeData.GetTree();
IEnumerable DataSource = TreeData.tree;
DataOperations operation = new DataOperations();
if (dm.Where != null && dm.Where.Count > 0)
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{
DataSource = operation.PerformFiltering(DataSource, dm.Where, "and");  //perform filtering
}
if (dm.Sorted != null && dm.Sorted.Count > 0)
{
DataSource = operation.PerformSorting(DataSource, dm.Sorted);  //perform sorting
}
var count = DataSource.ToList<TreeData>().Count();
if (dm.Skip != 0)
{
DataSource = operation.PerformSkip(DataSource, dm.Skip);   //Paging
}
if (dm.Take != 0)
{
DataSource = operation.PerformTake(DataSource, dm.Take);
}
if (dm.Where != null)
{
DataSource = CollectChildRecords(DataSource, dm);    // method to collect child records
}
return dm.RequiresCounts ? Json(new { result = DataSource, count = count }) : Json(DataSource);
}
public IEnumerable CollectChildRecords(IEnumerable datasource, DataManagerRequest dm)
{
DataOperations operation = new DataOperations();
IEnumerable DataSource = TreeData.tree;     // use the total DataSource here
string IdMapping = "TaskID";     // define your IdMapping field name here
int[] TaskIds = new int[0];
foreach (var rec in datasource)
{
int taskid = (int)rec.GetType().GetProperty(IdMapping).GetValue(rec);
TaskIds = TaskIds.Concat(new int[] { taskid }).ToArray();   //get the Parentrecord Ids based on IdMapping
Field
```

```
}

IEnumerable ChildRecords = null;

foreach (int id in TaskIds)

{

dm.Where[0].value = id;

IEnumerable records = operation.PerformFiltering(DataSource, dm.Where, dm.Where[0].Operator);
//perform filtering to collect the childrecords based on Ids

ChildRecords = ChildRecords == null || (ChildRecords.AsQueryable().Count() == 0) ? records :
((IEnumerable<object>)ChildRecords).Concat((IEnumerable<object>)records);  //concate the
childrecords with dataSource

}

if (ChildRecords != null)

{

ChildRecords = CollectChildRecords(ChildRecords, dm);    // repeat the operation for inner level child

if (dm.Sorted != null && dm.Sorted.Count > 0) // perform Sorting

{

ChildRecords = operation.PerformSorting(ChildRecords, dm.Sorted);

}

datasource = ((IEnumerable<object>)datasource).Concat((IEnumerable<object>)ChildRecords);
//concate the childrecords with dataSource

}

return datasource;

}
`
```

Offline mode

On remote data binding, all treegrid actions such as paging, loading child on-demand, will be processed on server-side. To avoid postback, set the treegrid to load all data on initialization and make the actions process in client-side. To enable this behavior, use the offline property of DataManager.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" idMapping='TaskID'
parentIdMapping='ParentItem' hasChildMapping='isParent' :treeColumnIndex='1'
:allowPaging='true' :pageSettings='pageSettings'>
            <e-columns>
                <e-column field='TaskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='TaskName' headerText='Task Name'
width=180></e-column>
```

```
                <e-column field='StartDate' headerText='Start Date' width=90
format="yMd" textAlign='Right'></e-column>
                <e-column field='Duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { DataManager, WebApiAdaptor } from "@syncfusion/ej2-data";
Vue.use(TreeGridPlugin);
export default {
  data () {
      let SERVICE_URI: string =
      "https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData";
    return {
      data: new DataManager({
        url: SERVICE_URI,
        adaptor: new WebApiAdaptor(),
        offline: true
      }),
      pageSettings: { pageCount: 4 }
    };
  },
  provide: {
      treegrid: [ Page ]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/data-binding/default-cs5" %}

Custom adaptor

You can create your own adaptor by extending the built-in adaptors. The following demonstrates custom adaptor approach and how to add a serial number for the records by overriding the built-in response processing using the `processResponse` method of the `WebApiAdaptor`.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" idMapping='TaskID'
parentIdMapping='ParentItem' hasChildMapping='isParent' :treeColumnIndex='1'
:allowPaging='true' :pageSettings='pageSettings'>
            <e-columns>
                <e-column field='Sno' headerText='SNO' width='90'
textAlign='Right'></e-column>
                <e-column field='TaskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='StartDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='Duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
```

```
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { DataManager, WebApiAdaptor } from "@syncfusion/ej2-data";
Vue.use(TreeGridPlugin);
class SerialNoAdaptor extends WebApiAdaptor {
    public processResponse(): Object[] {
        let i: number = 0;
        // calling base class processResponse function
        let original: Object[] = super.processResponse.apply(this,
arguments);
        // adding serial number
        original.forEach((item: Object) => item['Sno'] = ++i);
        return original;
    }
}
export default {
  data () {
      let SERVICE_URI: string =
      "https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData";
    return {
      data: new DataManager({
        url: SERVICE_URI,
        adaptor: new SerialNoAdaptor(),
        offline: true
      }),
      pageSettings: { pageCount: 4 }
    };
  },
  provide: {
      treegrid: [ Page ]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/data-binding/default-cs6" %}

Sending additional parameters to the server

To add a custom parameter to the data request, use the addParams method of Query class. Assign the Query object with additional parameters to the treegrid query property.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" idMapping='TaskID'
parentIdMapping='ParentItem' hasChildMapping='isParent' :query='query'
:treeColumnIndex='1' :allowPaging='true' :pageSettings='pageSettings'>
            <e-columns>
                <e-column field='TaskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
```

```
                <e-column field='TaskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='StartDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='Duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { DataManager, WebApiAdaptor , Query } from "@syncfusion/ej2-data";
Vue.use(TreeGridPlugin);
export default {
  data () {
      let SERVICE_URI: string =
      "https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData";
    return {
      data: new DataManager({
        url: SERVICE_URI,
        adaptor: new WebApiAdaptor(),
      }),
      pageSettings: { pageCount: 4 }
      query: new Query().addParams('ej2treegrid', 'true');
    };
  },
  provide: {
      treegrid: [ Page ]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/data-binding/default-cs7" %}

Handling HTTP error

During server interaction from the treegrid, some server-side exceptions may occur, and you can acquire those error messages or exception details in client-side using the actionFailure event.

The argument passed to the actionFailure event contains the error details returned from the server.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref=treegrid :dataSource="data" idMapping='TaskID'
parentIdMapping='parentItem':treeColumnIndex='1'
:actionFailure='actionFailure'>
            <e-columns>
                <e-column field='TaskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='TaskName' headerText='Task Name'
width=180></e-column>
                <e-column field='StartDate' headerText='Start Date' width=90
format="yMd" textAlign='Right'></e-column>
```

```
                <e-column field='Duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { DataManager, WebApiAdaptor } from "@syncfusion/ej2-data";
Vue.use(TreeGridPlugin);
export default {
  data () {
      let SERVICE_URI: string =
      "http://some.com/invalidUrl";
    return {
      data: new DataManager({
        url: SERVICE_URI,
        adaptor: new WebApiAdaptor(),
        offline: true
      }),
      pageSettings: { pageCount: 4 }
    };
  },
  provide: {
      treegrid: [ Page ]
    },
   methods:{
       actionFailure: function() {
       let span = document.createElement('span');
       let treegrid = document.getElementsByClassName("e-
treegrid")[0].ej2_instances[0];
       treegrid.element.parentNode.insertBefore(span, treegrid.element);
       span.style.color = '#FF0000'
       span.innerHTML = 'Server exception: 404 Not found';
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/data-binding/default-cs8" %}

The actionFailure event will be triggered not only for the server errors, but also when there is an exception while processing the treegrid actions.

Load on demand with virtualization

While binding remote data to Tree Grid component, by default Tree Grid renders parent rows in collapsed state. When expanding the root node, the child nodes will be loaded from the remote server.

When using virtualization with remote data binding, it helps you to improve the tree grid performance while loading a large set of data by setting enableVirtualization as true. The Tree Grid UI virtualization allows it to render only rows and columns visible within the view-port without buffering the entire datasource.hasChildMapping property maps the field name in data source, that denotes whether current record holds any child records. This is useful internally to show expand icon while binding child data on demand.

Copyright © 2001 -2024 Syncfusion Inc.

```
`

<template>

<div id="app">

<ejs-treegrid :dataSource='data' idMapping='TaskID' parentIdMapping='ParentValue'
hasChildMapping='isParent':treeColumnIndex='1' expandStateMapping='IsExpanded'
:enableVirtualization='true' height='400px' :allowSorting='true' :allowFiltering='true'
:editSettings='editSettings' :toolbar='toolbar'>

<e-columns>

<e-column field='TaskID'  :isPrimaryKey='true'  headerText='Task ID'  width=90 textAlign='Right'></e-
column>

<e-column field='TaskName' headerText='Task Name' width=180></e-column>

<e-column field='Duration' headerText='Duration' width=80></e-column>

</e-columns>

</ejs-treegrid>

</div>

</template>

<script>

import Vue from "vue";

import { TreeGridPlugin, VirtualScroll, Sort, Filter, Edit } from "@syncfusion/ej2-vue-treegrid";

import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';

Vue.use(TreeGridPlugin);

export default {

data() {

return {

data: new DataManager({

url: "Home/DataSource",

updateUrl: "Home/Update",

insertUrl: "Home/Insert",

removeUrl: "Home/Delete",

adaptor: new UrlAdaptor

}),

editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, newRowPosition: 'Below' },

toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],

};
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
provide: {
treegrid: [ Sort, VirtualScroll, Filter, Edit, Toolbar ]
}
}
</script>
```

The following code example describes handling of Load on demand at server end.

```ts
public ActionResult lazyLoading()
{
TreeData.tree = new List<TreeData>();
return View();
}
public ActionResult UrlDatasource(DataManagerRequest dm)
{
List<TreeData> data = new List<TreeData>();
data = TreeData.GetTree();
DataOperations operation = new DataOperations();
IEnumerable<TreeData> DataSource = data;
List<TreeData> ExpandedParentRecords = new List<TreeData>();
if (dm.Expand != null && dm.Expand[0] == "ExpandingAction") // setting the ExpandStateMapping property whether is true or false
{
var val = TreeData.GetTree().Where(ds => ds.TaskID == int.Parse(dm.Expand[1])).FirstOrDefault();
val.IsExpanded = true;
}
else if (dm.Expand != null && dm.Expand[0] == "CollapsingAction")
{
var val = TreeData.GetTree().Where(ds => ds.TaskID == int.Parse(dm.Expand[1])).FirstOrDefault();
val.IsExpanded = false;
}
if (!(dm.Where != null && dm.Where.Count > 1))
```

```
{
data = data.Where(p => p.ParentValue == null).ToList();
}
DataSource = data;
if (dm.Search != null && dm.Search.Count > 0) // Searching
{
DataSource = operation.PerformSearching(DataSource, dm.Search);
}
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // Sorting
{
DataSource = operation.PerformSorting(DataSource, dm.Sorted);
}
if (dm.Where != null && dm.Where.Count > 1) //filtering
{
DataSource = operation.PerformFiltering(DataSource, dm.Where, "and");
}
data = new List<TreeData>();
foreach (var rec in DataSource)
{
if (rec.IsExpanded)
{
ExpandedParentRecords.Add(rec as TreeData); // saving the expanded parent records
}
data.Add(rec as TreeData);
}
var GroupData = TreeData.GetTree().ToList().GroupBy(rec => rec.ParentValue)
.Where(g => g.Key != null).ToDictionary(g => g.Key?.ToString(), g => g.ToList());
if (ExpandedParentRecords.Count > 0)
{
foreach (var Record in ExpandedParentRecords.ToList())
{
var ChildGroup = GroupData[Record.TaskID.ToString()];
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // sorting the child records
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{
IEnumerable ChildSort = ChildGroup;
ChildSort = operation.PerformSorting(ChildSort, dm.Sorted);
ChildGroup = new List<TreeData>();
foreach (var rec in ChildSort)
{
ChildGroup.Add(rec as TreeData);
}
}
if (dm.Search != null && dm.Search.Count > 0) // searching the child records
{
IEnumerable ChildSearch = ChildGroup;
ChildSearch = operation.PerformSearching(ChildSearch, dm.Search);
ChildGroup = new List<TreeData>();
foreach (var rec in ChildSearch)
{
ChildGroup.Add(rec as TreeData);
}
}
AppendChildren(dm, ChildGroup, Record, GroupData, data);
}
}
DataSource = data;
if (dm.Expand != null && dm.Expand[0] == "CollapsingAction") // setting the skip index based on
collapsed parent
{
string IdMapping = "TaskID";
List<WhereFilter> CollapseFilter = new List<WhereFilter>();
CollapseFilter.Add(new WhereFilter() { Field = IdMapping, value = dm.Where[0].value, Operator =
dm.Where[0].Operator });
var CollapsedParentRecord = operation.PerformFiltering(DataSource, CollapseFilter, "and");
var index = data.Cast<object>().ToList().IndexOf(CollapsedParentRecord.Cast<object>().ToList()[0]);
dm.Skip = index;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
else if (dm.Expand != null && dm.Expand[0] == "ExpandingAction") // setting the skip index based on expanded parent
{
string IdMapping = "TaskID";
List<WhereFilter> ExpandFilter = new List<WhereFilter>();
ExpandFilter.Add(new WhereFilter() { Field = IdMapping, value = dm.Where[0].value, Operator = dm.Where[0].Operator });
var ExpandedParentRecord = operation.PerformFiltering(DataSource, ExpandFilter, "and");
var index = data.Cast<object>().ToList().IndexOf(ExpandedParentRecord.Cast<object>().ToList()[0]);
dm.Skip = index;
}
int count = data.Count;
DataSource = data;
if (dm.Skip != 0)
{
DataSource = operation.PerformSkip(DataSource, dm.Skip);   //Paging
}
if (dm.Take != 0)
{
DataSource = operation.PerformTake(DataSource, dm.Take);
}
return dm.RequiresCounts ? Json(new { result = DataSource, count = count }) : Json(DataSource);
}
private void AppendChildren(DataManagerRequest dm, List<TreeData> ChildRecords, TreeData ParentValue, Dictionary<string, List<TreeData>> GroupData, List<TreeData> data) // Getting child records for the respective parent
{
string TaskId = ParentValue.TaskID.ToString();
var index = data.IndexOf(ParentValue);
DataOperations operation = new DataOperations();
foreach (var Child in ChildRecords)
{
if (ParentValue.IsExpanded)
{
```

```
string ParentId = Child.ParentValue.ToString();
if (TaskId == ParentId)
{
((IList)data).Insert(++index, Child);
if (GroupData.ContainsKey(Child.TaskID.ToString()))
{
var DeepChildRecords = GroupData[Child.TaskID.ToString()];
if (DeepChildRecords?.Count > 0)
{
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // sorting the child records
{
IEnumerable ChildSort = DeepChildRecords;
ChildSort = operation.PerformSorting(ChildSort, dm.Sorted);
DeepChildRecords = new List<TreeData>();
foreach (var rec in ChildSort)
{
DeepChildRecords.Add(rec as TreeData);
}
}
if (dm.Search != null && dm.Search.Count > 0) // searching the child records
{
IEnumerable ChildSearch = DeepChildRecords;
ChildSearch = operation.PerformSearching(ChildSearch, dm.Search);
DeepChildRecords = new List<TreeData>();
foreach (var rec in ChildSearch)
{
DeepChildRecords.Add(rec as TreeData);
}
}
AppendChildren(dm, DeepChildRecords, Child, GroupData, data);
if (Child.IsExpanded)
{
index += DeepChildRecords.Count;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
}
}
else
{
Child.isParent = false;
}
}
}
}
}
public ActionResult Update(CRUDModel<TreeData> value)
{
List<TreeData> data = new List<TreeData>();
data = TreeData.GetTree();
var val = data.Where(ds => ds.TaskID == value.Value.TaskID).FirstOrDefault();
val.TaskName = value.Value.TaskName;
val.Duration = value.Value.Duration;
return Json(val);
}
public ActionResult Insert(CRUDModel<TreeData> value)
{
var c = 0;
for (; c < TreeData.GetTree().Count; c++)
{
if (TreeData.GetTree()[c].TaskID == value.RelationalKey)
{
if (TreeData.GetTree()[c].isParent == null)
{
TreeData.GetTree()[c].isParent = true;
}
break;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
c += FindChildRecords(value.RelationalKey);
TreeData.GetTree().Insert(c + 1, value.Value);
return Json(value.Value);
}
public int FindChildRecords(int? id)
{
var count = 0;
for (var i = 0; i < TreeData.GetTree().Count; i++)
{
if (TreeData.GetTree()[i].ParentValue == id)
{
count++;
count += FindChildRecords(TreeData.GetTree()[i].TaskID);
}
}
return count;
}
public object Delete(CRUDModel<TreeData> value)
{
if (value.deleted != null)
{
for (var i = 0; i < value.deleted.Count; i++)
{
TreeData.GetTree().Remove(TreeData.GetTree().Where(ds => ds.TaskID ==
value.deleted[i].TaskID).FirstOrDefault());
}
}
else
{
TreeData.GetTree().Remove(TreeData.GetTree().Where(or => or.TaskID ==
int.Parse(value.Key.ToString())).FirstOrDefault());
}
return Json(value);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
public class CRUDModel<T> where T : class
{
public TreeData Value;
public int Key { get; set; }
public int RelationalKey { get; set; }
public List<T> added { get; set; }
public List<T> changed { get; set; }
public List<T> deleted { get; set; }
}
public class TreeData
{
public static List<TreeData> tree = new List<TreeData>();
[System.ComponentModel.DataAnnotations.Key]
public int TaskID { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int? ParentValue { get; set; }
public bool? isParent { get; set; }
public bool IsExpanded { get; set; }
public TreeData() { }
public static List<TreeData> GetTree()
{
if (tree.Count == 0)
{
int root = 0;
for (var t = 1; t <= 500; t++)
{
Random ran = new Random();
string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ? "Release Breaker" : "Critical";
string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ? "Open" : "In Progress";
root++;
int rootItem = root;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
tree.Add(new TreeData() { TaskID = rootItem, TaskName = "Parent task " + rootItem.ToString(), isParent = true, IsExpanded = false, ParentValue = null, Duration = ran.Next(1, 50) });

int parent = root;

for (var d = 0; d < 1; d++)

{

root++;

string value = ((parent + 1) % 3 == 0) ? "Low" : "Critical";

int par = parent + 1;

progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";

int iD = root;

tree.Add(new TreeData() { TaskID = iD, TaskName = "Child task " + iD.ToString(), isParent = true, IsExpanded = false, ParentValue = rootItem, Duration = ran.Next(1, 50) });

int subparent = root;

for (var c = 0; c < 500; c++)

{

root++;

string val = ((subparent + c + 1) % 3 == 0) ? "Low" : "Critical";

int subchild = subparent + c + 1;

string progress = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";

int childID = root ;

tree.Add(new TreeData() { TaskID = childID, TaskName = "sub Child task " + childID.ToString(), isParent = false, IsExpanded = false, ParentValue = subparent, Duration = ran.Next(1, 50) });

}

}

}

}

return tree;

}

}

`
```

Load parent rows in expanded state with virtualization

Tree Grid provides an option to load the child records in the initial rendering itself for remote data binding by setting the loadChildOnDemand as true. When the loadChildOnDemand is enabled, parent records are rendered in expanded state.

Copyright © 2001 -2024 Syncfusion Inc.

When using virtualization with loadChildOnDemand , it helps you to improve the tree grid performance while loading the child records during the initial rendering for remote data binding by setting enableVirtualization as true and loadChildOnDemand as true.

```
<template>
<div id="app">
<ejs-treegrid :dataSource='data' idMapping='TaskID' parentIdMapping='ParentValue' hasChildMapping='isParent':treeColumnIndex='1' expandStateMapping='IsExpanded' :loadChildOnDemand='true' :enableVirtualization='true' height='400px' :allowSorting='true' :allowFiltering='true' :editSettings='editSettings' :toolbar='toolbar'>
<e-columns>
<e-column field='TaskID'  :isPrimaryKey='true'  headerText='Task ID'  width=90 textAlign='Right'></e-column>
<e-column field='TaskName' headerText='Task Name' width=180></e-column>
<e-column field='Duration' headerText='Duration' width=80></e-column>
</e-columns>
</ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, VirtualScroll, Sort, Filter, Edit } from "@syncfusion/ej2-vue-treegrid";
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
Vue.use(TreeGridPlugin);
export default {
data() {
return {
data: new DataManager({
url: "Home/DataSource",
updateUrl: "Home/Update",
insertUrl: "Home/Insert",
removeUrl: "Home/Delete",
adaptor: new UrlAdaptor
}),
editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, newRowPosition: 'Below' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
};
},
provide: {
treegrid: [ Sort, VirtualScroll, Filter, Edit, Toolbar ]
}
}
</script>
```

The following code example describes handling of child records at server end.

```ts
public ActionResult loadChildOndemand()
{
TreeData.tree = new List<TreeData>();
return View();
}
public ActionResult UrlDatasource(DataManagerRequest dm)
{
List<TreeData> data = new List<TreeData>();
data = TreeData.GetTree();
DataOperations operation = new DataOperations();
IEnumerable<TreeData> DataSource = data;
if (dm.Expand != null && dm.Expand[0] == "CollapsingAction") // setting the ExpandStateMapping property whether is true or false
{
var val = TreeData.GetTree().Where(ds => ds.TaskID == int.Parse(dm.Expand[1])).FirstOrDefault();
val.IsExpanded = false;
}
else if (dm.Expand != null && dm.Expand[0] == "ExpandingAction")
{
var val = TreeData.GetTree().Where(ds => ds.TaskID == int.Parse(dm.Expand[1])).FirstOrDefault();
val.IsExpanded = true;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
if (!(dm.Where != null && dm.Where.Count > 1))
{
data = data.Where(p => p.ParentValue == null).ToList();
}
DataSource = data;
if (dm.Search != null && dm.Search.Count > 0) // Searching
{
DataSource = operation.PerformSearching(DataSource, dm.Search);
}
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // Sorting
{
DataSource = operation.PerformSorting(DataSource, dm.Sorted);
}
if (dm.Where != null && dm.Where.Count > 1) //filtering
{
DataSource = operation.PerformFiltering(DataSource, dm.Where, "and");
}
data = new List<TreeData>();
foreach (var rec in DataSource)
{
data.Add(rec as TreeData);
}
var GroupData = TreeData.GetTree().ToList().GroupBy(rec => rec.ParentValue)
.Where(g => g.Key != null).ToDictionary(g => g.Key?.ToString(), g => g.ToList());
foreach (var Record in data.ToList())
{
if (GroupData.ContainsKey(Record.TaskID.ToString()))
{
var ChildGroup = GroupData[Record.TaskID.ToString()];
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // Sorting the child records
{
IEnumerable ChildSort = ChildGroup;
ChildSort = operation.PerformSorting(ChildSort, dm.Sorted);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
ChildGroup = new List<TreeData>();
foreach (var rec in ChildSort)
{
ChildGroup.Add(rec as TreeData);
}
}
if (dm.Search != null && dm.Search.Count > 0) // Searching the child records
{
IEnumerable ChildSearch = ChildGroup;
ChildSearch = operation.PerformSearching(ChildSearch, dm.Search);
ChildGroup = new List<TreeData>();
foreach (var rec in ChildSearch)
{
ChildGroup.Add(rec as TreeData);
}
}
if (ChildGroup?.Count > 0)
AppendChildren(dm, ChildGroup, Record, GroupData, data);
}
}
DataSource = data;
if (dm.Expand != null && dm.Expand[0] == "CollapsingAction") // setting the skip index based on collapsed parent
{
string IdMapping = "TaskID";
List<WhereFilter> CollapseFilter = new List<WhereFilter>();
CollapseFilter.Add(new WhereFilter() { Field = IdMapping, value = dm.Where[0].value, Operator = dm.Where[0].Operator });
var CollapsedParentRecord = operation.PerformFiltering(DataSource, CollapseFilter, "and");
var index = data.Cast<object>().ToList().IndexOf(CollapsedParentRecord.Cast<object>().ToList()[0]);
dm.Skip = index;
}
else if (dm.Expand != null && dm.Expand[0] == "ExpandingAction") // setting the skip index based on expanded parent
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{
string IdMapping = "TaskID";
List<WhereFilter> ExpandFilter = new List<WhereFilter>();
ExpandFilter.Add(new WhereFilter() { Field = IdMapping, value = dm.Where[0].value, Operator = dm.Where[0].Operator });
var ExpandedParentRecord = operation.PerformFiltering(DataSource, ExpandFilter, "and");
var index = data.Cast<object>().ToList().IndexOf(ExpandedParentRecord.Cast<object>().ToList()[0]);
dm.Skip = index;
}
int count = data.Count;
DataSource = data;
if (dm.Skip != 0)
{
DataSource = operation.PerformSkip(DataSource, dm.Skip);   //Paging
}
if (dm.Take != 0)
{
DataSource = operation.PerformTake(DataSource, dm.Take);
}
return dm.RequiresCounts ? Json(new { result = DataSource, count = count }) : Json(DataSource);
}
private void AppendChildren(DataManagerRequest dm, List<TreeData> ChildRecords, TreeData ParentValue, Dictionary<string, List<TreeData>> GroupData, List<TreeData> data) // Getting child records for the respective parent
{
string TaskId = ParentValue.TaskID.ToString();
var index = data.IndexOf(ParentValue);
DataOperations operation = new DataOperations();
foreach (var Child in ChildRecords)
{
if (ParentValue.IsExpanded)
{
string ParentId = Child.ParentValue.ToString();
if (TaskId == ParentId)
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{
((IList)data).Insert(++index, Child);
if (GroupData.ContainsKey(Child.TaskID.ToString()))
{
var DeepChildRecords = GroupData[Child.TaskID.ToString()];
if (DeepChildRecords?.Count > 0)
{
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // sorting the child records
{
IEnumerable ChildSort = DeepChildRecords;
ChildSort = operation.PerformSorting(ChildSort, dm.Sorted);
DeepChildRecords = new List<TreeData>();
foreach (var rec in ChildSort)
{
DeepChildRecords.Add(rec as TreeData);
}
}
if (dm.Search != null && dm.Search.Count > 0) // searching the child records
{
IEnumerable ChildSearch = DeepChildRecords;
ChildSearch = operation.PerformSearching(ChildSearch, dm.Search);
DeepChildRecords = new List<TreeData>();
foreach (var rec in ChildSearch)
{
DeepChildRecords.Add(rec as TreeData);
}
}
AppendChildren(dm, DeepChildRecords, Child, GroupData, data);
if (Child.IsExpanded)
{
index += DeepChildRecords.Count;
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
}
}
}
}
public ActionResult Update(CRUDModel<TreeData> value)
{
List<TreeData> data = new List<TreeData>();
data = TreeData.GetTree();
var val = data.Where(ds => ds.TaskID == value.Value.TaskID).FirstOrDefault();
val.TaskName = value.Value.TaskName;
val.Duration = value.Value.Duration;
return Json(val);
}
public ActionResult Insert(CRUDModel<TreeData> value)
{
var c = 0;
for (; c < TreeData.GetTree().Count; c++)
{
if (TreeData.GetTree()[c].TaskID == value.RelationalKey)
{
if (TreeData.GetTree()[c].isParent == null)
{
TreeData.GetTree()[c].isParent = true;
}
break;
}
}
c += FindChildRecords(value.RelationalKey);
TreeData.GetTree().Insert(c + 1, value.Value);
return Json(value.Value);
}
public int FindChildRecords(int? id)
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{
var count = 0;
for (var i = 0; i < TreeData.GetTree().Count; i++)
{
if (TreeData.GetTree()[i].ParentValue == id)
{
count++;
count += FindChildRecords(TreeData.GetTree()[i].TaskID);
}
}
return count;
}
public object Delete(CRUDModel<TreeData> value)
{
if (value.deleted != null)
{
for (var i = 0; i < value.deleted.Count; i++)
{
TreeData.GetTree().Remove(TreeData.GetTree().Where(ds => ds.TaskID ==
value.deleted[i].TaskID).FirstOrDefault());
}
}
else
{
TreeData.GetTree().Remove(TreeData.GetTree().Where(or => or.TaskID ==
int.Parse(value.Key.ToString())).FirstOrDefault());
}
return Json(value);
}
public class CRUDModel<T> where T : class
{
public TreeData Value;
public int Key { get; set; }
public int RelationalKey { get; set; }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
public List<T> added { get; set; }
public List<T> changed { get; set; }
public List<T> deleted { get; set; }
}
public class TreeData
{
public static List<TreeData> tree = new List<TreeData>();
[System.ComponentModel.DataAnnotations.Key]
public int TaskID { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int? ParentValue { get; set; }
public bool? isParent { get; set; }
public bool IsExpanded { get; set; }
public TreeData() { }
public static List<TreeData> GetTree()
{
if (tree.Count == 0)
{
int root = 0;
for (var t = 1; t <= 1500; t++)
{
Random ran = new Random();
string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ? "Release Breaker" : "Critical";
string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ? "Open" : "In Progress";
root++;
int rootItem = root;
tree.Add(new TreeData() { TaskID = rootItem, TaskName = "Parent task " + rootItem.ToString(), isParent
= true, IsExpanded = true, ParentValue = null, Duration = ran.Next(1, 50) });
int parent = root;
for (var d = 0; d < 1; d++)
{
root++;
```

```
string value = ((parent + 1) % 3 == 0) ? "Low" : "Critical";

int par = parent + 1;

progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";

int iD = root;

tree.Add(new TreeData() { TaskID = iD, TaskName = "Child task " + iD.ToString(), isParent = true,
IsExpanded = true, ParentValue = rootItem, Duration = ran.Next(1, 50) });

int subparent = root;

for (var c = 0; c < 6; c++)

{

root++;

string val = ((subparent + c + 1) % 3 == 0) ? "Low" : "Critical";

int subchild = subparent + c + 1;

string progress = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";

int childID = root ;

tree.Add(new TreeData() { TaskID = childID, TaskName = "sub Child task " + childID.ToString(),
ParentValue = subparent, Duration = ran.Next(1, 50) });

}

}

}

}

return tree;

}

}

`
```

Immutable mode in Vue Treegrid component

The immutable mode optimizes the TreeGrid re-rendering performance by using the object reference and deep compare concept. When performing the TreeGrid actions, it will only re-render the modified or newly added rows and prevent the re-rendering of the unchanged rows.

To enable this feature, you have to set the enableImmutableMode property as **true**.

* This feature uses the primary key value for data comparison. So, you need to provide the isPrimaryKey column.

APP.VUE

```
<template>
    <div id="app">
    <div id='container'>
        <table>
      <tbody>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <tr>
          <td>
            <span id="immutableDelete"></span>
          </td>
        </tr>
        <tr>
          <td>
            <span id="normalDelete"></span>
          </td>
        </tr>
        <tr>
          <td>
            <div>
              <ejs-button cssClass="e-info" v-
on:click.native="addTopEvent">Add row at top</ejs-button>
              <ejs-button cssClass="e-info" v-
on:click.native="addBottomEvent">Add row at bottom</ejs-button>
              <ejs-button cssClass="e-info" v-
on:click.native="deleteEvent">Delete 5 rows</ejs-button>
              <ejs-button cssClass="e-info" v-
on:click.native="sortEvent">Sort Task ID</ejs-button>
              <ejs-button cssClass="e-info" v-
on:click.native="pageEvent">Paging</ejs-button>
            </div>
          </td>
        </tr>
        <tr>
          <td>
        <span><b>Immutable TreeGrid</b></span>
        <ejs-treegrid ref="immutablegrid" :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1' height=350 :allowPaging='true'
:pageSettings='pageSettings' :editSettings='editSettings'
:selectionSettings='selectionOptions' :enableImmutableMode='true'
:beforeDataBound="immutableBeforeDataBound" :dataBound="immutableDataBound">
        <e-columns>
                <e-column field='taskID' :isPrimaryKey='true'
headerText='Task ID' width='90' textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='200'></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' width='90' format='yMd' type='date'></e-column>
                <e-column field='endDate' headerText='End Date'
textAlign='Right' width='90' format='yMd' type='date'></e-column>
                <e-column field='duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
                <e-column field='priority' headerText='Priority' width='90'
textAlign='Right'></e-column>
                <e-column field='approved' headerText='Approved' width='90'
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
          </td>
          <td>
            <span><b>Normal TreeGrid</b></span>
        <ejs-treegrid ref="nomalgrid" :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1' height=350 :allowPaging='true'
:pageSettings='pageSettings' :editSettings='editSettings'
```

```
:selectionSettings='selectionOptions' :enableImmutableMode='true'
:beforeDataBound="beforeDataBound" :dataBound="dataBound">
        <e-columns>
                <e-column field='taskID' :isPrimaryKey='true'
headerText='Task ID' width='90' textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='200'></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' width='90' format='yMd' type='date'></e-column>
                <e-column field='endDate' headerText='End Date'
textAlign='Right' width='90' format='yMd' type='date'></e-column>
                <e-column field='duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
                <e-column field='priority' headerText='Priority' width='90'
textAlign='Right'></e-column>
                <e-column field='approved' headerText='Approved' width='90'
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
          </td>
        </tr>
      </tbody>
    </table>
    </div>
  </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Edit } from "@syncfusion/ej2-vue-treegrid";
import { sampleData, sampleData2, dataSource1 } from "./datasource.js";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(TreeGridPlugin);
Vue.use(ButtonPlugin);
if(sampleData2.length == 0){
  dataSource1();
}
export default {
  data() {
    return {
      data: sampleData2,
      pageSettings: { pageSize: 50 },
      selectionOptions: { type: 'Multiple' },
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Cell' },
      immutableStart: 0,
      normalStart: 0,
      immutableInit: true,
      init: true
    };
  },
  provide: {
    treegrid: [Page, Edit],
  },
  methods: {
    addTopEvent: function () {
      let immutableInstance = this.$refs.immutablegrid;
      let instance = this.$refs.nomalgrid;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      immutableInstance.addRecord(sampleData[0], 0, "Top");
      instance.addRecord(sampleData[0], 0, "Top");
    },
    addBottomEvent: function () {
      let immutableInstance = this.$refs.immutablegrid;
      let instance = this.$refs.nomalgrid;
      immutableInstance.addRecord(sampleData[0], 0, "Bottom");
      instance.addRecord(sampleData[0], 0, "Bottom");
    },
    deleteEvent: function () {
      let immutableInstance = this.$refs.immutablegrid;
      let instance = this.$refs.nomalgrid;
      immutableInstance.selectRows([0, 2, 4]);
      instance.selectRows([0, 2, 4]);
      immutableInstance.deleteRecord();
      instance.deleteRecord();
    },
    sortEvent: function () {
      let immutableInstance = this.$refs.immutablegrid;
      let instance = this.$refs.nomalgrid;
      let aData = immutableInstance.dataSource.reverse();
      immutableInstance.setProperties({ dataSource: aData });
      instance.setProperties({ dataSource: aData });
    },
    pageEvent: function () {
      let immutableInstance = this.$refs.immutablegrid;
      let instance = this.$refs.nomalgrid;
      let page = instance.ej2Instances.pageSettings.currentPage + 1;
      immutableInstance.goToPage(page);
      instance.goToPage(page);
    },
    immutableBeforeDataBound: function () {
      this.immutableStart = new Date().getTime();
    },
    immutableDataBound: function () {
      let val = this.immutableInit ? '' : new Date().getTime() -
this.immutableStart;
      document.getElementById("immutableDelete").innerHTML =
        "Immutable rendering Time: " + "<b>" + val + "</b>" + "<b>ms</b>";
      this.immutableStart = 0; this.immutableInit = false;
    },
    beforeDataBound: function () {
      this.normalStart = new Date().getTime();
    },
    dataBound: function () {
      let val = this.init ? '' : new Date().getTime() - this.normalStart;
      document.getElementById("normalDelete").innerHTML =
        "Normal rendering Time: " + "<b>" + val + "</b>" + "<b>ms</b>";
      this.normalStart = 0; this.init = false;
    },
  },
};
</script>
<style>
.e-treegrid {
  pointer-events: none;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
 @import "../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/immutable-mode/default-cs1" %}

Limitations

The following features are not supported in the immutable mode:

- Frozen rows and columns
- Row Template
- Detail Template
- Column reorder
- Virtualization

Columns

Columns in Vue Treegrid component

The column definitions are used as the dataSource schema in the TreeGrid. This plays a vital role in rendering column values in the required format.

The treegrid operations such as sorting, filtering and searching etc. are performed based on column definitions. The field property of the columns

is necessary to map the data source values in TreeGrid columns.

1. If the column field is not specified in the dataSource, the column values will be empty.

2. If the field name contains “dot” operator, it is considered as complex binding.

treeColumnIndex property denotes the column that is used to expand and collapse child rows.

Format

To format cell values based on specific culture, use the columns.format property. The TreeGrid uses Internalization library to format number and date

values.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' height='315px'>
            <e-columns>
                <e-column field='orderID' headerText='Order ID' width=90
textAlign='Right'></e-column>
                <e-column field='orderName' headerText='Order Name'
width=220></e-column>
                <e-column field='price' headerText='Price' format='C2'
width=80 textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
```

```
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { formatData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: formatData,
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs10" %}

By default, the number and date values are formatted in en-US locale.

Number formatting

The number or integer values can be formatted using the below format strings.

Format |Description |Remarks

{ type:'date', format:'dd/MM/yyyy' } | 04/07/1996

{ type:'date', format:'dd.MM.yyyy' } | 04.07.1996

{ type:'date', skeleton:'short' } | 7/4/96

{ type: 'dateTime', format: 'dd/MM/yyyy hh:mm a' } | 04/07/1996 12:00 AM

{ type: 'dateTime', format: 'MM/dd/yyyy hh:mm:ss a' } | 07/04/1996 12:00:00 AM

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' height='315px'>
            <e-columns>
                <e-column field='orderID' headerText='Order ID' width=90
textAlign='Right'></e-column>
                <e-column field='orderName' headerText='Order Name'
width=100></e-column>
                <e-column field='orderDate' headerText='Order Date'
width=160 :format='formatOptions' type='date' textAlign='Right'></e-column>
                <e-column field='price' headerText='Price' width=90
format='C2' textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { formatData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      data: formatData,
      formatOptions: {type:'date', format:'dd/MM/yyyy'},
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs11" %}

Lock columns

You can lock columns by using column.lockColumn property. The locked columns will be moved to the first position. Also you can't reorder its position.

In the below example, duration column is locked and its reordering functionality is disabled.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' :allowReordering='true' :allowSelection='false'
height='315px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText='Start Date' width=90
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right' :lockColumn='true'
:customAttributes="customAttributes"></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Reorder } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      customAttributes : {class: 'customcss'}
    };
  },
  provide: {
      treegrid: [ Reorder ]
    },
}
</script>
<style>
 @import
"https://ej2.syncfusion.com/vue/documentation/node_modules/@syncfusion/ej2-
vue-treegrid/styles/material.css";
```

```
  .e-grid .e-rowcell.customcss{
   background-color: #ecedee;
}
.e-grid .e-headercell.customcss{
   background-color: #ecedee;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs12" %}

Touch interaction

When the right edge of the header cell is tapped, a floating handler will be visible over the right border of the column. To resize the column, tap and drag the floating handler as needed.

The following screenshot represents the column resizing in touch device.

<!-- markdownlint-disable MD033 -->

<img src="../images/column-resizing.png" alt="Touch interaction image" style="width:320px;height:620px">

<!-- markdownlint-enable MD033 -->

Column type

Column type can be specified using the columns.type property. It specifies the type of data the column binds.

If the format is defined for a column, the column uses type to select the appropriate format option (number

or date).

TreeGrid column supports the following types:

- string
- number
- boolean
- date
- datetime

If the type is not defined, it will be determined from the first record of the dataSource.

Checkbox column

To render checkboxes in existing column, you need to set [columns.showCheckbox] property as true.

It is also possible to select the rows hierarchically using checkboxes in TreeGrid by enabling [autoCheckHierarchy] property. When we check on any parent record checkbox then the child record checkboxes will get checked.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex='1'
:autoCheckHierarchy='true' childMapping='subtasks' :height='400'>
            <e-columns>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name' width=180
:showCheckbox='true'></e-column>
                <e-column field='startDate' headerText=' Start Date'
textAlign='Right' format='yMd' type='date' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right'  width=80></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
 data() {
    return {
      data: sampleData,
    };
  },
  provide: {
      treegrid: [Page]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs13" %}

Controlling Tree Grid actions

You can enable or disable treegrid action for a particular column by setting the allowFiltering, and allowSorting properties.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' :allowFiltering="true" :allowSorting="true"
:height='270'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=90
:allowSorting="false" textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText='Start Date' width=90
:allowFiltering="false" format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { TreeGridPlugin, Sort , Filter } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
 data() {
    return {
      data: sampleData
    };
  },
  provide: {
      treegrid: [Sort, Filter]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs14" %}

Show/hide columns by external button

You can show or hide treegrid columns dynamically using external buttons by invoking the showColumns or hideColumns method.

APP.VUE

```
<template>
<div id="app">
        <ejs-button @click.native="show"> Show </ejs-button>
        <ejs-button @click.native="hide"> Hide </ejs-button>
        <ejs-treegrid ref=treegrid :dataSource="data"
childMapping='subtasks' :treeColumnIndex='1' :height='270'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText='Start Date' width=90
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(TreeGridPlugin);
Vue.use(ButtonPlugin);
export default {
  data() {
    return {
      data: sampleData
    };
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
   methods:{
       show: function() {
        this.$refs.treegrid.showColumns(['Task ID', 'Duration']); // show by
HeaderText
    },
    hide: function() {
        this.$refs.treegrid.hideColumns(['Task ID', 'Duration']); // hide by
HeaderText
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs15" %}

ValueAccessor

The valueAccessor is used to access/manipulate the value of display data. You can achieve custom value formatting by using the valueAccessor.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid  :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' height='315px'>
            <e-columns>
                <e-column field='orderID' headerText='Order ID' width=100
textAlign='Right'></e-column>
                <e-column field='orderName' headerText='Order Name'
:valueAccessor='orderFormatter' width=215></e-column>
                <e-column field='orderDate' headerText='Order Date'
width=110 :format='formatOptions' type='date' textAlign='Right'></e-column>
                <e-column field='price' headerText='Price' width=100
:valueAccessor='currencyFormatter' textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { formatData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
   data() {
    return {
      data: formatData,
      formatOptions: {type:'date', format:'dd/MM/yyyy'},
    };
    },
    methods:{
        currencyFormatter: function(field, data, column) {
            return '€' + data['price'];
        },
    orderFormatter: function (field, data, column) {
```

```
        return data[field] + '-' + data['Category'];
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs16" %}

Display array type columns

You can bind an array of objects in a column by using the valueAccessor property.

In this example, the name field has an array of two objects, FirstName and LastName. These two objects are joined and bound to a column using the

valueAccessor.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid  :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='315px'>
            <e-columns>
               <e-column field='taskID' headerText='Task ID' width=110
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='name' headerText='Assignee' width=150
:valueAccessor='valueAccess' textAlign='Left'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { stringData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: stringData,
    };
    },
    methods:{
          valueAccess: function(field, data, column) {
              return data[field].map((s) => { return s.lastName ||
s.firstName; }).join(' ');
              }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs17" %}

Copyright © 2001 -2024 Syncfusion Inc.

Expression column

You can achieve the expression column by using the valueAccessor property.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid  :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' height='315px'>
            <e-columns>
                <e-column field='orderID' headerText='Order ID' width=110
textAlign='Right'></e-column>
                <e-column field='orderName' headerText='Order Name'
width=210></e-column>
                <e-column field='units' headerText='Units' width=120
textAlign='Right'></e-column>
                <e-column field='unitPrice' headerText='Unit Price'
width=120 textAlign='Right'></e-column>
                <e-column field='price' headerText='Price' width=120
:valueAccessor='totalPrice'
                textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { formatData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: formatData,
    };
    },
    methods:{
        totalPrice: function (field, data, column) {
            return data.units * data.unitPrice;
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs18" %}

How to render boolean values as checkbox

To render boolean values as checkbox in columns, you need to set displayAsCheckBox property as true.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' height='315px'>
            <e-columns>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-column field='taskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='approved' headerText='Approved' width=90
displayAsCheckBox='true'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs19" %}

Headers in Vue Treegrid component

Header text

By default, column header title is displayed from column field value. To override the default header title, you have to define the headerText value.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' height='315px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText='Start Date' width=90
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs21" %}

* If both the field and headerText

are not defined in the column, the column renders with “empty” header text.

Header template

You can customize the header element by using the headerTemplate property.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='0' height='315px'>
            <e-columns>
                <e-column field='taskName' :headerTemplate='projectName'
width=180></e-column>
                <e-column field='startDate' :headerTemplate='dateTemplate'
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration'
:headerTemplate='durationTemplate' textAlign='Right'></e-column>
                <e-column field='progress'
:headerTemplate='progressTemplate' textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { headerData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: headerData,
       projectName: function () {
          return { template : Vue.component('columnTemplate',{
             template: `<div class="image">
                    <img :src="image" width=20 height=20 class="e-image"/>
Task Name
                </div>`,
                data: function() {
                    return {
                        data: {}
                    }
```

```
                },
                computed: {
                    image: function() {
                        return "Taskname.png";
                    }
                }
            })}
        },
      dateTemplate: function () {
            return { template : Vue.component('columnTemplate',{
               template: `<div class="image">
                    <img :src="image" width=20 height=20 class="e-image"/>
Start Date
                </div>`,
                data: function() {
                    return {
                        data: {}
                    }
                },
                computed: {
                    image: function() {
                        return "Startdate.png";
                    }
                }
            })}
        },
        durationTemplate: function () {
            return { template : Vue.component('columnTemplate',{
               template: `<div class="image">
                    <img :src="image" width=20 height=20 class="e-image"/>
Duration
                </div>`,
                data: function() {
                    return {
                        data: {}
                    }
                },
                computed: {
                    image: function() {
                        return "Duration.png";
                    }
                }
            })}
        },
        progressTemplate: function () {
            return { template : Vue.component('columnTemplate',{
               template: `<div class="image">
                    <img :src="image" width=20 height=20 class="e-image"/>
Progress
                </div>`,
                data: function() {
                    return {
                        data: {}
                    }
                },
                computed: {
                    image: function() {
```

```
                        return "progress.png" ;
                    }
                }
            })}
        }
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs22" %}

Column template in Vue Treegrid component

The column template has options to display custom element instead of a field value in the column.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex='0' :height='260'
:rowHeight='83' childMapping='Children'>
            <e-columns>
                <e-column field='EmpID' headerText='Employee ID'
width=95></e-column>
                <e-column field='Name' headerText='Name' width=110></e-
column>
                <e-column field='DOB' headerText=' Start Date'
textAlign='Right' format='yMd' width=90></e-column>
                <e-column field='Year GR' textAlign='Center'
:template='template1'  width=120></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { SparklinePlugin } from "@syncfusion/ej2-vue-charts";
import { sparkdata, getSparkData, textData } from "./datasource.js";
import { RowDataBoundEventArgs, getObject } from '@syncfusion/ej2-grids';
import { EmitType } from '@syncfusion/ej2-base';
import { Sparkline, ISparklineLoadEventArgs, SparklineTheme } from
'@syncfusion/ej2-charts';
Vue.use(TreeGridPlugin);
Vue.use(SparklinePlugin);
let winloss: Sparkline;
export default {
 data() {
    return {
      data: textData,
       template1: function () {
          return { template : Vue.component('columnTemplate',{
             template: `<ejs-sparkline :id='elemid' height='50px'
width='150px' type='WinLoss' valueType='Numeric' fill='#3C78EF'
negativePointColor='#f7a816' tiePointColor='darkgray'
:dataSource='sparkData(data.EmployeeID)'></ejs-sparkline> `,
```

```
                    data: function() {
                        return {
                            data: {},
                        }
                    },
                    methods:{
                        sparkData: function(id: number){
                            return getSparkData('column', id);
                        }
                    },
                    computed: {
                        elemid: function() {
                            return 'spkwl' + this.data.EmployeeID;
                         },
            })}
    };
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/column-template/default-cs1" %}

TreeGrid actions such as editing, filtering and sorting etc. will depend upon the column field. If the field is not specified in the template column, the treegrid actions cannot be performed.

Using condition template

You can render the template elements based on condition.

In the following code, checkbox is rendered based on Approved field value.

`

<script id="template" type="text/x-template">

`

<input type="checkbox" checked />

<div v-else class="template_checkbox">

<input type="checkbox" />

`

</script>

`

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' height='315px' >
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='approved' headerText='Approved' width=120
textAlign='Centre' :template='approvedtemplate'></e-column>
                <e-column field='progress' headerText='Progress' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
 data() {
    return {
      data: sampleData,
      approvedtemplate: function () {
          return { template : Vue.component('columnTemplate',{
             template: `<div v-if=cData class="template_checkbox">
                    <input type="checkbox" checked />
                </div>
                <div v-else class="template_checkbox">
                    <input type="checkbox" />
                </div>`,
                data: function() {
                    return {
                        data: {}
                    }
                },
                computed: {
                    cData: function() {
                        return this.data.approved;
                    }
                }
          })}
    };
  }
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs9" %}

Complex data binding in Vue Treegrid component

You can achieve complex data binding in the treegrid by using the dot(.) operator in the column.field.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :height='315'>
            <e-columns>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-column field='taskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='assignee.firstName' headerText='Assignee'
width=90></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { complexData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
   data() {
    return {
      data: complexData
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs20" %}

Column reorder in Vue Treegrid component

Reordering can be done by drag and drop of a particular column header from one index to another index within the treegrid. To enable reordering, set the allowReordering to true.

To use reordering, inject the Reorder module in the treegrid.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:allowReordering='true' :treeColumnIndex='1' height='285px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText='Start Date' width=90
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Reorder } from "@syncfusion/ej2-vue-treegrid";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
    };
  },
   provide: {
      treegrid: [ Reorder ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs4" %}

You can disable reordering a particular column by setting the columns.allowReordering to false.

Reorder multiple columns

Multiple columns can be reordered at a time by using the reorderColumns method.

APP.VUE

```
<template>
<div id="app">
    <ejs-button id='reorderMultipleCols' @click.native='reorder'>Reorder
TASK ID AND DURATION TO LAST</ejs-button>
        <ejs-treegrid ref='treegrid' :dataSource="data"
childMapping='subtasks' :allowReordering='true' :treeColumnIndex='1'
height='285px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                <e-column field='progress' headerText='Progress' width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Reorder } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
Vue.use(TreeGridPlugin);
Vue.use(ButtonPlugin);
export default {
  data() {
    return {
      data: sampleData,
    };
  },
   provide: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      treegrid: [ Reorder ]
    },
     methods:{
       reorder: function() {

this.$refs.treegrid.reorderColumns(['taskID','duration'],'progress');
      }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs5" %}

Column resizing in Vue Treegrid component

Column width can be resized by clicking and dragging the right edge of the column header. While dragging, the width of the respective column will be resized immediately. Each column can be auto resized by double-clicking the right edge of the column header to fit the width of that column based on the widest cell content. To enable column resize, set the allowResizing property to true.

To use the column resize, inject Resize module in the treegrid.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' :allowResizing='true' height='315px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText='Start Date' width=90
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Resize } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
    };
  },
   provide: {
      treegrid: [ Resize ]
    },
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs6" %}

* You can disable resizing for a particular column by setting the columns.allowResizing to false.

* In RTL mode, you can click and drag the left edge of the header cell to resize the column.

Min and max width

Column resize can be restricted between minimum and maximum width by defining the columns->minWidth and columns->maxWidth.

In the following sample, minimum and maximum width are defined for Duration, and Task Name columns.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' :allowResizing='true' height='315px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'minWidth=
170 width=180 maxWidth=250></e-column>
                <e-column field='duration' headerText='Duration' minWidth=
50 width=80 maxWidth=150 textAlign='Right'></e-column>
                <e-column field='progress' headerText='Progress'  width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Resize } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
    };
  },
   provide: {
      treegrid: [ Resize ]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs7" %}

Resize stacked column

Stacked columns can be resized by clicking and dragging the right edge of the stacked column header. While dragging, the width of the respective child columns will be resized at the same time. You can disable resize for particular stacked column by setting allowResizing as false to its columns.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' :allowResizing='true' height='260px'>
             <e-columns>
                <e-column headerText='Order Details'
:columns='orderColumns'></e-column>
                <e-column headerText='Shipment Details'
:columns='shipColumns'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Resize } from "@syncfusion/ej2-vue-treegrid";
import { stackedData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: stackedData,
        orderColumns : [
            { field: 'orderID', headerText: 'Order ID', textAlign: 'Right',
width: 90 },
            { field: 'orderName', headerText: 'Order Name', textAlign:
'Left', width: 170 },
        ],
        shipColumns : [
            { field: 'shipMentCategory', headerText: 'Shipment Category',
textAlign: 'Left', width: 90 },
            { field: 'shippedDate', headerText: 'Shipped Date', textAlign:
'Right', format: 'yMd', width: 90 }
        ]
    };
  },
   provide: {
      treegrid: [ Resize ]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs8" %}

Column menu in Vue Treegrid component

The column menu has options to integrate features like sorting, filtering, and autofit. It will show a menu with the integrated feature when users click on multiple icon of the column. To enable column menu, you need to define the showColumnMenu property as true.

By default, column menu is enabled for all columns and you can disable column menu for a particular column by defining the showColumnMenu as false in columns property.

To use the column menu, inject the ColumnMenu module in the treegrid.

Copyright © 2001 -2024 Syncfusion Inc.

The default items are displayed in following table.

| Item | Description |

|-----|-----|

| SortAscending | Sort the current column in ascending order. |

| SortDescending | Sort the current column in descending order. |

| AutoFit | Auto fit the current column. |

| AutoFitAll | Auto fit all columns. |

| Filter | Show the filter option as given in filterSettings.type |

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex='1'
:allowPaging='true' :showColumnMenu='true' childMapping='subtasks'
         :allowFiltering='true' :filterSettings='filterSettings'
:allowResizing='true' :pageSettings='pageSettings'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText=' Start Date'
textAlign='Right' :showColumnMenu='false' format='yMd' type='date'
width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Sort, Resize, ColumnMenu, Page, Filter } from
"@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
 data() {
    return {
      data: sampleData,
      filterSettings: { type: "Menu" },
      pageSettings: { pageCount: 4, pageSize: 10 }
    };
  },
  provide: {
      treegrid: [Sort, Resize, ColumnMenu, Page, Filter]
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs1" %}

Custom column menu item

The custom column menu items can be added to the column menu by defining the columnMenuItems as a collection of the MenuItemModel.

The action for custom column menu items can be performed using columnMenuClick event.

In the below example, clear sorting action was performed through columnMenuItems by using clearSorting method of the Tree Grid.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid':dataSource="data" :treeColumnIndex='1'
:allowPaging='true' :allowSorting='true' :sortSettings='sortSettings'
:showColumnMenu='true' childMapping='subtasks'
:columnMenuItems='columnMenuItems' :columnMenuClick='columnMenuClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText=' Start Date'
textAlign='Right' format='yMd' type='date' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right'  width=80></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Sort, ColumnMenu, Page } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
 data() {
    return {
      data: sampleData,
      columnMenuItems: [{text:'Clear Sorting', id:'clearsorting'}],
      sortSettings: {columns:[{direction: "Ascending", field: "taskID"}]}
    };
  },
  provide: {
      treegrid: [Sort, ColumnMenu, Page]
  },
  methods: {
  columnMenuClick: function(args) {
      if(args.item.id === 'clearsorting'){
        this.$refs.treegrid.clearSorting();
      }
    }
  },
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs2" %}

Customize menu items for particular columns

It is possible to customize specific items from the column menu for particular columns using columnMenuOpen event. columnMenuOpen event can be utilized to determine which items to customize in column menu.

The following example shows how to hide the built-in **Filter** menu item when the column menu is opened for the **taskName** column, while allowing it to remain visible for all other columns.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid':dataSource="data" :treeColumnIndex='1'
:allowPaging='true' :allowSorting='true' :showColumnMenu='true'
childMapping='subtasks' :columnMenuOpen='columnMenuOpen'
:allowFiltering='true' :filterSettings='filterSettings'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText=' Start Date'
textAlign='Right' format='yMd' type='date' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right'  width=80></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Sort, ColumnMenu, Page, Filter} from
"@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
 data() {
    return {
      data: sampleData,
      filterSettings: { type: "Menu" },
    };
  },
  provide: {
      treegrid: [Sort, ColumnMenu, Page, Filter]
  },
  methods: {
   columnMenuOpen: function (args) {
      for (let item of args.items) {
        if (item.text === 'Filter' && args.column.field === 'taskName') {
            item.hide = true;
        } else {
            item.hide = false;
```

```
        }
      }
    }
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs3" %}

Responsive columns in Vue Treegrid component

You can toggle column visibility based on media queries which are defined

at the hideAtMedia.

The hideAtMedia accepts valid

Media Queries.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex='1'
childMapping='subtasks' height='315px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right'  hideAtMedia='(min-width: 700px)' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText=' Start Date'
textAlign='Right'  hideAtMedia='(max-width: 500px)' format='yMd' type='date'
width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/columns/default-cs23" %}

Row

Copyright © 2001 -2024 Syncfusion Inc.

Row in Vue Treegrid component

The row represents record details fetched from data source.

Customize rows

You can customize the appearance of a row by using the rowDataBound event.

The rowDataBound event triggers for every row. In the event handler, you can get the

RowDataBoundEventArgs that contains details of the row.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex='1'
childMapping='subtasks' height='315px' :rowDataBound='rowDataBound'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right'  width='90'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText=' Start Date'
textAlign='Right'  format='yMd' type='date' width='90'></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right'  width='80'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: sampleData,
    };
  },
  methods:{
     rowDataBound: function(args) {
         if (args.data['duration'] == 0 ) {
        args.row.style.background= '#336c12';
    } else if (args.data['duration'] < 3) {
        args.row.style.background= '#7b2b1d';
    }
    }
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/row/default-cs1" %}

Styling alternate rows

You can change the treegrid's alternative rows' background color by overriding the .e-altrow class.

Copyright © 2001 -2024 Syncfusion Inc.

`

.e-treegrid .e-altrow {

background-color: #fafafa;

}

`

Please refer to the following example.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex='1'
childMapping='subtasks' height='315px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right'  width='90'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText=' Start Date'
textAlign='Right'  format='yMd' type='date' width='90'></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right'  width='80'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: sampleData,
    };
  },
}
</script>
<style>
  .e-grid .e-altrow {
    background-color: #fafafa;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/row/default-cs2" %}

Row height in Vue Treegrid component

You can customize the row height of treegrid rows through the rowHeight property. The rowHeight property is used to change the row height of entire treegrid rows.

In the below example, the rowHeight is set as '60px'.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex='1'
childMapping='subtasks' :enableHover='false' :allowSelection='false'
height='275' rowHeight=60>
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right'  width='90'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText=' Start Date'
textAlign='Right'  format='yMd' type='date' width='90'></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right'  width='80'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: sampleData,
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/row/row-cs4" %}

Customize row height for particular row

Grid row height for particular row can be customized using the rowDataBound

event by setting the rowHeight in arguments for each row based on the requirement.

In the below example, the row height for the row with Task ID as '3' is set as '90px' using the rowDataBound event.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex='1'
childMapping='subtasks' :rowDataBound='rowDataBound' height='275' >
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right'  width='90'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText=' Start Date'
textAlign='Right'  format='yMd' type='date' width='90'></e-column>
```

```
                <e-column field='duration' headerText='Duration'
textAlign='Right'  width='80'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: sampleData,
    };
  },
  methods: {
   rowDataBound: function (args) {
     if((args.data as TasKDetails).taskID === 3){
        args.rowHeight = 90;
      }
    }
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/row/row-cs5" %}

Row template in Vue Treegrid component

The rowTemplate has an option to customise the look and behavior of the treegrid rows. The rowTemplate property accepts either the template string or HTML element ID.

APP.VUE

```
<template>
<div id="app">
<ejs-treegrid :dataSource="data" childMapping='Children' height=275
:rowHeight = '83' width='auto' :treeColumnIndex='0'
:rowTemplate='rowTemplate'>
            <e-columns>
                <e-column field='EmpID' headerText='Employee ID'
width='180'></e-column>
                <e-column field='Name' headerText='Employee Name' width =
'150'></e-column>
                <e-column field='Address' headerText = 'Employee Details'
width='350'></e-column>
            </e-columns>
        </ejs-treegrid>
        </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { textdata } from "./datasource.js";
Vue.use(TreeGridPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data: () => {
    return {
      data: textdata,
      rowTemplate: function () {
        return { template : Vue.component('rowTemplate',{
        template: `<tr>
    <td class="border" style='padding-left:18px;'>
        <div>{{data.EmpID}}</div>
                        </td>
                        <td class="border" style='padding: 10px 0px 0px
20px;'>
                            <div style="font-size:14px;">
                              {{data.Name}}
                                <p style="font-
size:9px;">{{data.Designation}}</p>
                            </div>
                        </td>
                        <td class="border">
                            <div>
                                <div
style="position:relative;display:inline-block;">
                                    <img :src="image" :alt="data.FullName"
/>
                                </div>
                                <div style="display:inline-block;">
                                    <div
style="padding:5px;">{{data.Address}}</div>
                                    <div
style="padding:5px;">{{data.Country}}</div>
                                    <div style="padding:5px;font-
size:12px;">{{data.Contact}}</div>
                                </div>
                            </div>
                        </td>
                </tr>`,
                data: function() {
            return {
              data: {}
            }
          },
          computed: {
    image: function() {
      return '../../../../treegrid/images/' + this.data.FullName + '.png';
    }
  }
  })}
      }
    };
  }
}
</script>
<style>
 @import "../../node_modules/@syncfusion/ej2-vue-
treegrid/styles/material.css";
  .border {
        border-color: #e0e0e0;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        border: 1px solid #e0e0e0;
        border-width: 1px 0px 0px 0px;
    }
    img {
        width: 60px;
        height: 60px;
        vertical-align: baseline;
        border-radius: 50px;
        margin-left: 20px;
        background-repeat: no-repeat;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/row/rowtemplate-cs1" %}

The rowTemplate property accepts only the TR element.

Row template with formatting

If the rowTemplate is used, the value cannot be formatted inside the template using the columns.format property. In that case, a function should be defined globally to format the value and invoke it inside the template.

APP.VUE

```
<template>
<div id="app">
  <ejs-treegrid :dataSource="data" childMapping='Children' height=275
:rowHeight = '83' width='auto' :treeColumnIndex='0'
:rowTemplate='rowTemplate'>
            <e-columns>
                <e-column field='EmpID' headerText='Employee ID'
width='180'></e-column>
                <e-column field='Address' headerText = 'Employee Details'
width='350'></e-column>
                <e-column field='DOB' headerText = 'DOB' width = '150'></e-
column>
            </e-columns>
        </ejs-treegrid>
        </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { textdata } from "./datasource.js";
import { Internationalization } from '@syncfusion/ej2-base';
let instance = new Internationalization();
Vue.use(TreeGridPlugin);
export default {
  data: () => {
    return {
      data: textdata,
      rowTemplate: function () {
        return { template : Vue.component('rowTemplate',{
        template: `<tr>
    <td class="border" style='padding-left:18px;'>
        <div>{{data.EmpID}}</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                        </td>
                        <td class="border">
                            <div>
                                <div
style="position:relative;display:inline-block;">
                                    <img :src="image" :alt="data.FullName"
/>
                                </div>
                                <div style="display:inline-block;">
                                    <div
style="padding:5px;">{{data.Address}}</div>
                                    <div
style="padding:5px;">{{data.Country}}</div>
                                    <div style="padding:5px;font-
size:12px;">{{data.Contact}}</div>
                                </div>
                            </div>
                        </td>
                        <td class="border" style='padding-left: 20px;'>
                            <div>{{format(data.DOB)}}</div>
                        </td>
                </tr>`,
                data: function() {
            return {
              data: {}
            }
          },
          methods: {
    format: function(value) {
        return instance.formatDate(value, { skeleton: 'yMd', type: 'date'
});
    }
  },
    computed: {
    image: function() {
      return '../../../../treegrid/images/' + this.data.FullName + '.png';
    }
  }
  })}
      }
    };
  }
}
</script>
<style>
 @import "../../node_modules/@syncfusion/ej2-vue-
treegrid/styles/material.css";
  .border {
        border-color: #e0e0e0;
        border: 1px solid #e0e0e0;
        border-width: 1px 0px 0px 0px;
    }
    img {
        width: 60px;
        height: 60px;
        vertical-align: baseline;
        border-radius: 50px;
```

```
        margin-left: 20px;
        background-repeat: no-repeat;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/row/rowtemplateformat-cs1" %}

Limitations

Row template feature is not compatible with all the features which are available in treegrid and it has limited features support. Here we have listed out the features which are not compatible with row template feature.

- Filtering
- Paging
- Sorting
- Scrolling
- Searching
- Rtl
- Context Menu
- State Persistence

Detail template in Vue Treegrid component

The detail template provides additional information about a particular row. By expanding the parent row the child rows are expanded along with their detail template. The detailTemplate property accepts either the template string or HTML element ID.

APP.VUE

```
<template>
<div id="app">
<ejs-treegrid :dataSource="data" childMapping='Children' height=317
width='auto' :treeColumnIndex='0' :detailTemplate='detailTemplate'>
    <e-columns>
      <e-column field='Name' headerText='First Name' width='180'></e-column>
      <e-column field='DOB' headerText = 'DOB' width='105' type='date'
format='yMd'></e-column>
      <e-column field='Designation' headerText = 'Designation'
width='180'></e-column>
      <e-column field='Country' headerText = 'Country' width='148'></e-
column>
    </e-columns>
</ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, DetailRow } from "@syncfusion/ej2-vue-treegrid";
import { textdata } from "./datasource.js";
import { Internationalization } from '@syncfusion/ej2-base';
let instance = new Internationalization();
Vue.use(TreeGridPlugin);
export default {
  data: () => {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    return {
      data: textdata,
      detailTemplate: function () {
        return { template : Vue.component('detailTemplate',{
        template: `<div>
     <div style="position: relative; display: inline-block; float: left;
font-weight: bold; width: 10%;padding:5px 4px 2px 27px;;">
        <img :src="image" alt="{{data.FullName}}"/>
    </div>
    <div style="padding-left: 10px; display: inline-block; width: 66%; text-
wrap: normal;font-size:13px;font-family:'Segoe UI';">
        <div class="e-description" style="word-wrap: break-word;">
            <b>{{data.Name}}</b> was born on {{format(data.DOB)}}. Now lives
at {{data.Address}}, {{data.Country}}. {{data.Name}} holds a position of
<b>{{data.Designation}}</b> in our WA department, (Seattle USA).
        </div>
        <div class="e-description" style="word-wrap: break-word;margin-
top:5px;">
            <b style="margin-right:10px;">Contact:</b>{{data.Contact}}
        </div>
    </div>
</div>`,
    data: function() {
            return {
              data: {}
            }
          },
          methods: {
    format: function(value) {
        return instance.formatDate(value, { skeleton: 'yMd', type: 'date'
});
    }
  },
    computed: {
    image: function() {
       return '../../../../treegrid/images/' + this.data.FullName + '.png';
    }
  }
  })}
      }
    };
  },
  provide: {
  treegrid: [DetailRow]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/row/detailtemplate-cs1" %}

Row drag and drop in Vue Treegrid component

The TreeGrid rows can be reordered, dropped to another TreeGrid or custom control by enabling the allowRowDragAndDrop to true.

To use row drag and drop, inject the RowDD module in the TreeGrid.

Copyright © 2001 -2024 Syncfusion Inc.

Drag and drop within Tree Grid

The TreeGrid row drag and drop allows you to drag and drop TreeGrid rows on the same TreeGrid using drag icon. To enable row drag and drop, set the allowRowDragAndDrop to true. It provides the way to drop the row above, below or child to the target row with respective to the target row position.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid id='TreeGrid' :dataSource="data"
:allowRowDragAndDrop='true' height='315' :treeColumnIndex='1'
:rowDropSettings='rowDrop'
         :selectionSettings='selectionSettings'
         childMapping='subtasks' >
            <e-columns>
                    <e-column field='taskID' headerText='Task ID'
:isPrimaryKey='true' textAlign='Right' width=90></e-column>
                    <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                    <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
                    <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, RowDD } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: sampleData,
      selectionSettings: { type: 'Multiple' },
      rowDrop: { targetID: 'destTree' }
    };
  },
    provide: {
    treegrid: [RowDD]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/row/row-cs1" %}

* Selection feature must be enabled for row drag and drop.

* For multiple row selection, the type property must be set to multiple.

* The isPrimaryKey property is necessary to perform row drag and drop operation.

Copyright © 2001 -2024 Syncfusion Inc.

Drag and drop to another Tree Grid

To drag and drop between two TreeGrid, enable the allowRowDragAndDrop property and specify the target TreeGrid ID in targetID property of rowDropSettings.

APP.VUE

```
<template>
<div id="app">
<div style="display: inline-block">
        <ejs-treegrid id='TreeGrid' :dataSource="data"
:allowRowDragAndDrop='true' height='315' :treeColumnIndex='1'
:rowDropSettings='rowDrop' :selectionSettings='selectionSettings'
childMapping='subtasks' >
            <e-columns>
                <e-column field="taskID" headerText="Task ID" width="120"
isPrimaryKey="true"></e-column>
                <e-column field="taskName" headerText="Task Name"
width="220"></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
                <e-column field="duration" headerText="Duration"
width="90"></e-column></e-columns>
        </ejs-treegrid>
        <ejs-treegrid id='DestTree' :allowRowDragAndDrop='true' height='315'
:treeColumnIndex='1' :rowDropSettings='rowDrops'
:selectionSettings='selectionSettings' childMapping='subtasks' >
            <e-columns>
                <e-column field="taskID" headerText="Task ID" width="120"
isPrimaryKey="true"></e-column>
                <e-column field="taskName" headerText="Task Name"
width="220"></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
                <e-column field="duration" headerText="Duration"
width="90"></e-column>
            </e-columns>
        </ejs-treegrid>
        </div>
</div>
</template>
<script>
    import Vue from "vue";
    import {TreeGridPlugin, RowDD, Selection} from "@syncfusion/ej2-vue-
treegrid";
    import { sampleData } from "./datasource.js";
    Vue.use(TreeGridPlugin);
    export default {
        data() {
            return {
                data: sampleData,
                selectionSettings: { type: 'Multiple' },
                rowDrop: { targetID: 'TreeGrid' },
                rowDrops: { targetID: 'DestTree' }
            };
        },
        provide: {
            treegrid: [RowDD, Selection]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        }
    }
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/row/row-cs2" %}

Drag and drop events

The following events are triggered while drag and drop the treegrid rows.

rowDragStartHelper - Triggers when click the drag icon or treegrid row and this event is used to customize the drag element based on user criteria.

rowDragStart -Triggers when starts to drag the treegrid row.

rowDrag - Triggers while dragging the treegrid row.

rowDrop - Triggers when a drag element is dropped on the target element.

Prevent reordering a row as child to another row

You can prevent the default behavior of dropping rows as children to the target by setting the cancel property to true in rowDrop event argument. You can also change the drop position after cancelling using reorderRows method.

In the below example drop action is cancelled and dropped above to target row.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid id='TreeGrid' ref='treeGrid' :dataSource="data"
:allowRowDragAndDrop='true' height='315' :treeColumnIndex='1'
         childMapping='subtasks' :rowDrop="rowDrop" >
            <e-columns>
                    <e-column field='taskID' headerText='Task ID'
:isPrimaryKey='true' textAlign='Right' width=90></e-column>
                    <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                    <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
                    <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, RowDD } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: sampleData,
    };
  },
    provide: {
```

```
    treegrid: [RowDD]
  },
  methods: {
    rowDrop: function(args) {
              if (args.dropPosition == 'middleSegment') {
                    args.cancel = true;
                    this.$refs.treeGrid.reorderRows([args.fromIndex],
args.dropIndex, 'above');
                  }
            }
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/row/row-cs3" %}

Indent in Vue Treegrid component

The Indent and Outdent feature will help to change the hierarchy level of rows in tree grid. The indent action moves the selected row as the last child of its previous row, whereas the outdent action moves the selected row as a sibling to its parent row.

To use the indent and outdent feature, inject the RowDD module in the Tree Grid. The tree grid toolbar has the built-in items to execute indent and outdent actions. Define this by using the toolbar property.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :selectedRowIndex='2' height='270px'
:toolbar='toolbar'>
        <e-columns>
                <e-column field='taskID'  :isPrimaryKey='true'
headerText='Task ID'  width=90 textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'  width=180
></e-column>
                <e-column field='priority' headerText='Priority' width=80
textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Toolbar, RowDD } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      toolbar: ['Indent', 'Outdent']
    };
```

```
  },
    provide: {
    treegrid: [ Toolbar, RowDD ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/indent-outdent/default-cs1" %}

Indent/Outdent a row programmatically

You can change the hierarchy level of record programmatically using indent and outdent methods.

APP.VUE

```
<template>
<div id="app">
        <button v-on:click="Indenting()">Indent</button>
        <button v-on:click="Outdenting()">Outdent</button>
        <ejs-treegrid :dataSource='data' ref="treegrid"
childMapping='subtasks' :treeColumnIndex='1' height='270px'>
        <e-columns>
                <e-column field='taskID'  :isPrimaryKey='true'
headerText='Task ID'  width=90 textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'  width=180
></e-column>
                <e-column field='priority' headerText='Priority' width=80
textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, RowDD } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData
    };
  },
  methods: {
    Indenting() {

this.$refs.treegrid.indent(this.$refs.treegrid.getCurrentViewRecords()[2]);
    }
    Outdenting() {

this.$refs.treegrid.outdent(this.$refs.treegrid.getCurrentViewRecords()[2]);
    }
  },
    provide: {
    treegrid: [ RowDD ]
```

```
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/programmatic-indent/default-cs1" %}

Cell

Content in Vue Treegrid component

The HTML tags can be displayed in the TreeGrid header and content by enabling the disableHtmlEncode property.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='300px'>
        <e-columns>
        <e-column field='taskID' headerText='<span> Task ID </span>'
:disableHtmlEncode='true' width=140 textAlign='Right'></e-column>
        <e-column field='taskName' headerText='<span> Task Name </span>'
:disableHtmlEncode='false' width=180 ></e-column>
        <e-column field='startDate' headerText='Start Date' width=80
format="yMd" textAlign='Right'></e-column>
        <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
        </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { htmlData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: htmlData,
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/cell/default-cs6" %}

Auto wrap in Vue Treegrid component

The auto wrap allows the cell content of the treegrid to wrap to the next line when it exceeds the boundary of the cell width. The Cell Content wrapping works based on the position of white space between words. To enable auto wrap, set the allowTextWrap property to true. You can configure the auto wrap mode by setting the textWrapSettings.wrapMode property.

There are three types of wrapMode. They are:

Copyright © 2001 -2024 Syncfusion Inc.

- **Both**: Both value is set by default. Auto wrap will be enabled for both the content and the header.
- **Header**: Auto wrap will be enabled only for the header.
- **Content**: Auto wrap will be enabled only for the content.

Note: When a column width is not specified, then auto wrap of columns will be adjusted with respect to the treegrid's width.

In the following example, the textWrapSettings.wrapMode is set to Content.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='300px' :allowTextWrap='true'
:textWrapSettings='wrapSettings'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=90
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=75></e-column>
                <e-column field='startDate' headerText='Start Date' width=90
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      wrapSettings: { wrapMode: 'Content' }
    };
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/cell/default-cs1" %}

Clip mode in Vue Treegrid component

The clip mode provides options to display its overflow cell content and it can be defined by the columns.clipMode property.

There are three types of clipMode. They are:

- **Clip**: Truncates the cell content when it overflows its area.
- **Ellipsis**: Displays ellipsis when the cell content overflows its area.

Copyright © 2001 -2024 Syncfusion Inc.

- **EllipsisWithTooltip**: Displays ellipsis when the cell content overflows its area, also it will display the tooltip while hover on ellipsis is applied.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='300px' gridLines='Both'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=90
textAlign='Right' ></e-column>
                <e-column field='taskName' headerText='Task Name' width=90
clipMode='Clip'></e-column>
                <e-column field='assignee.firstName' headerText='Assignee'
clipMode='EllipsisWithTooltip' width=40 textAlign='Right'></e-column>
                 <e-column field='priority' headerText='Priority'
clipMode='Ellipsis' width=40 textAlign='Right'></e-column>
                 <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { complexData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: complexData,
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/cell/default-cs5" %}

By default, columns.clipMode value is Ellipsis.

Editing

Edit in Vue Treegrid component

The TreeGrid component has options to dynamically insert, delete and update records. Editing feature is enabled by using editSettings property and it requires a primary key column for CRUD operations.

To define the primary key, set columns.isPrimaryKey to true in particular column.

To use CRUD, inject the Link to the Video module in treegrid.

To get start quickly with Edit Options, you can check on this video:

APP.VUE

```
<template>
```

```
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='270px' :editSettings='editSettings'>
        <e-columns>
                <e-column field='taskID'  :isPrimaryKey='true'
headerText='Task ID'  width=90 textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'  width=180
></e-column>
                <e-column field='priority' headerText='Priority' width=80
textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true }
    };
  },
    provide: {
    treegrid: [ Edit]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs7" %}

* You can disable editing for a particular column, by specifying

columns.allowEditing to false.

Toolbar with edit option

The treegrid toolbar has the built-in items to execute Editing actions.

You can define this by using the toolbar property.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='270px' :editSettings='editSettings'
:toolbar='toolbar'>
        <e-columns>
                <e-column field='taskID'  :isPrimaryKey='true'
headerText='Task ID'  width=90 textAlign='Right'></e-column>
```

```
                <e-column field='taskName' headerText='Task Name'  width=180
></e-column>
                <e-column field='priority' headerText='Priority' width=80
textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true,  mode: 'Row' },
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel']
    };
  },
    provide: {
    treegrid: [ Edit, Toolbar ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs8" %}

<!-- markdownlint-disable MD033 -->

<!-- Template context

The Essential JS2 packages has a built-in template engine. Using the template engine, you can access the row information inside the HTML element and bind the attributes, values, or elements based on this row information.

The following properties will be available at the time of template execution.

| Property Name | Usage |

|---------------|--------|

| isAdd | A Boolean property; it defines whether the current row should be a new record or not.

In the following code example, the OrderID textbox has been disabled by using the isAdd property.

`

// The disabled attributes will be added based on the isAdd property.

<input id="taskID" name="taskID" v-model='data.taskID' type="text" :disabled="!data.isAdd">

Copyright © 2001 -2024 Syncfusion Inc.

`

The following code example illustrates rendering the taskID textbox, when a new record is added.

`

${if(isAdd)}

<input id="taskID" name="taskID" v-model='data.taskID' type="text" :disabled="!data.isAdd">

<label class="e-float-text e-label-top" for="OrderID">Task ID</label>

${/if}

`

The dialog template syntax supports the ES6 expression string literals, and you can refer to the Template Engine for more template syntax.

Render editors as components

You can convert the form editors to EJ2 controls in the actionComplete event based on the requestType as beginEdit or add.

The following code example illustrates rendering the drop-down list control in the actionComplete event.

`

<script>

actionComplete: function(args) {

if ((args.requestType === 'beginEdit' || args.requestType === 'add')) {

let priorityData: {}[] = DataUtil.distinct(this.$refs.treegrid.grid.dataSource, 'priority',true);

new DropDownList({value: args.rowData.priority, popupHeight: '200px', floatLabelType: 'Always',

dataSource: priorityData, fields: {text: 'priority', value: 'priority'}, placeholder: 'Priority'},
args.form.elements.namedItem('priority') as HTMLInputElement);

}

}

</script>

`

Get value from editor

Copyright © 2001 -2024 Syncfusion Inc.

You can read, format, and update the current editor value in the actionBegin event at the time of setting requestType to save.

In the following code example, the progress value has been formatted and updated.

` html

<script>

actionBegin: function(args) {

if (args.requestType === 'save') {

// cast string to integer value.

args.data['progress'] = parseFloat(args.form.querySelector("#progress").value);

}

}

</script>

`

Set focus to editor

By default, the first input element in the dialog will be focused while opening the dialog.

If the first input element is in disabled or hidden state, focus the valid input element in the

actionComplete

event based on requestType as beginEdit.

`

<script>

actionComplete: function(args) {

// Set initail Focus

if (args.requestType === 'beginEdit') {

(args.form.elements.namedItem('taskName')).focus();

}

}

</script>

`

Adding validation rules for custom editors

If you have used additional fields that are not present in the column model, then add the validation rules to the actionComplete event.

`

<script>

actionComplete: function(args) {

Copyright © 2001 -2024 Syncfusion Inc.

if ((args.requestType === 'beginEdit' || args.requestType === 'add')) {

// Add Validation Rules

args.form.ej2_instances[0].addRules('progress', {max: 100});

}

}

</script>

`

-->

<!-- markdownlint-disable MD033 -->

Adding row position

The TreeGrid control provides the support to add the new row in the top, bottom, above selected row, below selected row and child position of tree grid content using editSettings.newRowPosition property. By default, a new row will be added at the top of the treegrid.

The following examples shows how to set new row position as Child in tree grid.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='270px' :editSettings='editSettings'
:toolbar='toolbar'>
        <e-columns>
                <e-column field='taskID'  :isPrimaryKey='true'
headerText='Task ID'  width=90 textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
editType= 'datepickeredit' type= 'date' format= 'yMd' textAlign='Right'
width=170></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true,  mode: 'Cell',  newRowPosition: 'Child' },
      toolbar: ['Add', 'Delete', 'Update', 'Cancel']
```

```
    };
  },
    provide: {
    treegrid: [ Edit, Toolbar ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs9" %}

Confirmation messages

Delete confirmation

The delete confirm dialog can be shown when deleting a record by defining the showDeleteConfirmDialog as true

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1' height='270px'
:editSettings='editSettings' :toolbar='toolbar'>
        <e-columns>
        <e-column field='taskID'  :isPrimaryKey='true'  headerText='Task ID'
width=90 textAlign='Right'></e-column>
        <e-column field='taskName' headerText='Task Name' width=180></e-
column>
        <e-column field='startDate' headerText='Start Date' editType=
'datepickeredit' type= 'date' format='yMd' textAlign='Right' width=170></e-
column>
        <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
        </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Cell', showDeleteConfirmDialog: true },
    };
  },
    provide: {
    treegrid: [ Edit, Toolbar ]
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs10" %}

The showDeleteConfirmDialog supports all type of edit modes.

Default column values on add new

The treegrid provides an option to set the default value for the columns when adding a new record in it.

To set a default value for the particular column by defining the columns.defaultValue.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='270px' :editSettings='editSettings'
:toolbar='toolbar'>
        <e-columns>
        <e-column field='taskID' :isPrimaryKey='true'  headerText='Task ID'
width=90 textAlign='Right'></e-column>
        <e-column field='taskName' headerText='Task Name' width=180></e-
column>
        <e-column field='startDate' headerText='Start Date' editType=
'datepickeredit' type='date' format='yMd' textAlign='Right' width=170></e-
column>
        <e-column field='priority' headerText='Priority' width=110
defaultValue='Normal'></e-column>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Cell', showDeleteConfirmDialog: true },
    };
  },
    provide: {
    treegrid: [ Edit, Toolbar ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs11" %}

Disable editing for particular column

You can disable editing for particular columns by using the columns.allowEditing.

Copyright © 2001 -2024 Syncfusion Inc.

In the following demo, editing is disabled for the Start Date column.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='270px' :editSettings='editSettings'
:toolbar='toolbar'>
        <e-columns>
        <e-column field='taskID'  :isPrimaryKey='true'  headerText='Task ID'
width=90 textAlign='Right'></e-column>
        <e-column field='taskName' headerText='Task Name' width=180></e-
column>
        <e-column field='startDate' headerText='Start Date'
:allowEditing='false' :allowEditing='false' type='date' format= 'yMd'
textAlign='Right' width=170></e-column>
        <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
        </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Cell' },
    };
  },
    provide: {
    treegrid: [ Edit, Toolbar ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs12" %}

Troubleshoot: Editing works only for first row

The Editing functionalities can be performed based upon the primary key value of the selected row. If primaryKey is not defined in the treegrid, then edit or delete action take places the first row.

Edit types in Vue Treegrid component

Cell edit type and its params

The columns.editType is used to customize the edit type of the particular column.

You can set the columns.editType based on data type of the column.

Copyright © 2001 -2024 Syncfusion Inc.

- numericedit - NumericTextBox component for integers, double, and decimal data types.
- defaultedit - TextBox component for string data type.
- dropdownedit - DropDownList component for list data type.
- booleanedit - CheckBox component for boolean data type.
- datepickeredit - DatePicker component for date data type.
- datetimepickeredit - DateTimePicker component for date time data type.

Also, you can customize model of the columns.editType component through the columns.edit.params.

The following table describes cell edit type component and their corresponding edit params of the column.

Component |Example

NumericTextBox | params: { decimals: 2, value: 5 }

TextBox | -

DropDownList | params: { value: 'Germany' }

Checkbox | params: { checked: true}

DatePicker | params: { format:'dd.MM.yyyy' }

DateTimePicker | params: { value: new Date() }

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='270px' :editSettings='editSettings'
:toolbar='toolbar'>
        <e-columns>
                <e-column field='taskID'  :isPrimaryKey='true'
headerText='Task ID'  width=90 textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name' editType=
'stringedit' width=170 ></e-column>
                <e-column field='startDate' headerText='Start Date'
editType= 'datetimepickeredit' :edit='dateParams' :format='formatOptions'
textAlign='Right' width=170></e-column>
                <e-column field='approved' headerText='Approved' editType=
'booleanedit' type='boolean' width=110 displayAsCheckBox='true'
textAlign='Right'></e-column>
                <e-column field='progress' headerText='Progress' width=120
editType= 'numericedit' :edit='progressParams' textAlign='Right'></e-column>
                 <e-column field='priority' headerText='Priority' editType=
'dropdownedit'  width=110 textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true,  mode: 'Row' },
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
      formatOptions: {type:'dateTime', format:'M/d/y hh:mm a'},
      dateParams: { params: {format: 'M/d/y hh:mm a' } },
      progressParams: { params: {format: 'n' } },
    };
  },
    provide: {
    treegrid: [ Edit, Toolbar]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs6" %}

If edit type is not defined in the column, then it will be considered as the stringedit type (Textbox component).

Cell editing in Vue Treegrid component

In Cell edit mode, when you double click on a cell, it is changed to edit state. You can change the cell value and save to the data source.

To enable Cell edit, set the editSettings.mode as Cell.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1' height='270px'
:editSettings='editSettings' :toolbar='toolbar'>
        <e-columns>
                <e-column field='taskID'  :isPrimaryKey='true'
headerText='Task ID'  width=90 textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
editType= 'datepickeredit' type= 'date' format= 'yMd' textAlign='Right'
width=170></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Cell' },
    };
  },
    provide: {
    treegrid: [ Edit, Toolbar ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs2" %}

Cell edit mode is default mode of editing.

Row editing in Vue Treegrid component

In Row edit mode, when you start editing the currently selected record, the entire row is changed to edit state.

You can change the cell values of the row and save edited data to the data source.

To enable Row edit, set the editSettings.mode as Row.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1' height='270px'
:editSettings='editSettings' :toolbar='toolbar'>
        <e-columns>
                <e-column field='taskID'  :isPrimaryKey='true'
headerText='Task ID'  width=90 textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
editType= 'datepickeredit' type= 'date' format= 'yMd' textAlign='Right'
width=170></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    return {
      data: sampleData,
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Row' },
    };
  },
    provide: {
    treegrid: [ Edit, Toolbar ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs13" %}

Dialog editing in Vue Treegrid component

In Dialog edit mode, when you start editing the currently selected row, data will be shown on a dialog. You can change the cell values and save edited data to the data source.

To enable Dialog edit, set the editSettings.mode as Dialog.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1' height='270px'
:editSettings='editSettings' :toolbar='toolbar'>
        <e-columns>
                <e-column field='taskID'  :isPrimaryKey='true'
headerText='Task ID'  width=90 textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
editType= 'datepickeredit' type= 'date' format= 'yMd' textAlign='Right'
width=170></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog' },
    };
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
    provide: {
    treegrid: [ Edit, Toolbar ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs5" %}

Batch editing in Vue Treegrid component

In Batch edit mode, when you double-click on the tree grid cell, then the target cell changed to edit state. You can bulk save (added, changed and deleted data in the single request) to data source by click on the toolbar's **Update** button or by externally invoking the batchSave method. To enable Batch edit, set the editSettings.mode as **Batch**.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' idMapping='TaskID'
parentIdMapping='parentID' :treeColumnIndex='1' height='270px'
:editSettings='editSettings' :toolbar='toolbar'>
        <e-columns>
                <e-column field='TaskID'  :isPrimaryKey='true'
headerText='Task ID'  width=90 textAlign='Right'></e-column>
                <e-column field='TaskName' headerText='Task Name'
width=180></e-column>
                <e-column field='StartDate' headerText='Start Date'
editType= 'datepickeredit' type= 'date' format= 'yMd' textAlign='Right'
width=170></e-column>
                <e-column field='Duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { projectData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: projectData,
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Batch' },
    };
  },
    provide: {
    treegrid: [ Edit, Toolbar ]
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
 @import "../../node_modules/@syncfusion/ej2-vue-grids/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs1" %}

Command column editing in Vue Treegrid component

The command column provides an option to add CRUD action buttons in a column. This can be defined by the column.commands property.

The available built-in command buttons are:

| Command Button | Actions |

|----------------|---------|

| Edit | Edit the current row.|

| Delete | Delete the current row.|

| Save | Update the edited row.|

| Cancel | Cancel the edited state. |

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='270px' :editSettings='editSettings'>
        <e-columns>
               <e-column field='taskID'  :isPrimaryKey='true'
headerText='Task ID'  width=90 textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width=180></e-column>
                <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
                <e-column  headerText='Manage Records' :commands='commands'
width=80  textAlign='Right'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, CommandColumn } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true,  mode: 'Row' },
       commands: [{ type: 'Edit', buttonOption: { cssClass: 'e-flat',
iconCss: 'e-edit e-icons' } },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        { type: 'Delete', buttonOption: { cssClass: 'e-flat', iconCss: 'e-
delete e-icons' } },
        { type: 'Save', buttonOption: { cssClass: 'e-flat', iconCss: 'e-
update e-icons' } },
        { type: 'Cancel', buttonOption: { cssClass: 'e-flat', iconCss: 'e-
cancel-icon e-icons' } }]
    };
  },
    provide: {
    treegrid: [ Edit, CommandColumn ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs3" %}

Custom command

The custom command buttons can be added in a column by using the column.commands property and the action for the custom buttons can be defined in the buttonOption.click event.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1' height='270px'
:editSettings='editSettings'>
        <e-columns>
        <e-column field='taskID'  :isPrimaryKey='true'  headerText='Task ID'
width=90 textAlign='Right'></e-column>
        <e-column field='taskName' headerText='Task Name' width=180></e-
column>
        <e-column field='duration' headerText='Duration' width=80
textAlign='Right'></e-column>
        <e-column  headerText='Commands' :commands='commands'
width=120></e-column>
        </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, CommandColumn, IRow, Column } from
"@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
import { closest } from '@syncfusion/ej2-base';
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true,  mode: 'Row' },
       commands: [{ buttonOption: { content: 'Details', cssClass: 'e-flat',
click: this.onClick } }]
    };
```

```
  },
    provide: {
    treegrid: [ Edit, CommandColumn ]
  },
    methods: {
    onClick: function(args) {
    let rowIndex: number = (closest(args.target as Element, '.e-
row')).rowIndex;
    let data: Object =
this.$refs.treegrid.ej2Instances.getCurrentViewRecords();
    alert(JSON.stringify(data[rowIndex]));
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs4" %}

Validation in Vue Treegrid component

Column validation

Column validation allows you to validate the edited or added row data and it display errors for invalid fields before saving data.

TreeGrid uses Form Validator component for column validation. You can set validation rules by defining the columns.validationRules.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='270px' :editSettings='editSettings'
:toolbar='toolbar'>
        <e-columns>
                <e-column field='taskID'  :isPrimaryKey='true'
headerText='Task ID'  width=90 textAlign='Right'
:validationRules='taskIDRules'></e-column>
                <e-column field='taskName' headerText='Task Name' editType=
'stringedit' :validationRules='taskNameRules' width=170 ></e-column>
                <e-column field='startDate' headerText='Start Date'
editType= 'datetimepickeredit' :validationRules='dateRules'
:edit='dateParams' :format='formatOptions' textAlign='Right' width=170></e-
column>
                <e-column field='approved' headerText='Approved' editType=
'booleanedit' type='boolean' width=110 displayAsCheckBox='true'
textAlign='Right'></e-column>
                <e-column field='progress' headerText='Progress' width=120
editType= 'numericedit' :validationRules='progressRules'
:edit='progressParams' textAlign='Right'></e-column>
                <e-column field='priority' headerText='Priority' editType=
'dropdownedit'  width=110 textAlign='Right'
:validationRules='priorityRules'></e-column>
                </e-columns>
        </ejs-treegrid>
</div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true,  mode: 'Cell', showDeleteConfirmDialog: true },
      toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
      taskIDRules: { required: true, number: true },
      taskNameRules: { required: true },
      dateRules: { date: true }
      formatOptions: {type:'dateTime', format:'M/d/y hh:mm a'},
      dateParams: { params: {format: 'M/d/y hh:mm a' } },
      progressRules: { min: 0 , number: true },
      progressParams: { params: {format: 'n' } },
      priorityRules: { required: true }
    };
  },
    provide: {
    treegrid: [ Edit, Toolbar ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs14" %}

Custom validation

You can define your own custom validation rules for the specific columns by using Form Validator custom rules.

In the below demo, custom validation applied for Priority column.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='270px' :editSettings='editSettings'
:toolbar='toolbar'>
        <e-columns>
        <e-column field='taskID'  :isPrimaryKey='true'  headerText='Task ID'
width=90 textAlign='Right'></e-column>
        <e-column field='taskName' headerText='Task Name' width=180></e-
column>
        <e-column field='startDate' headerText='Start Date' editType=
'datepickeredit' type= 'date' format= 'yMd' textAlign='Right' width=170></e-
column>
        <e-column field='priority' headerText='Priority' width=110
:validationRules='priorityRules'></e-column>
        </ejs-treegrid>
</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Cell', showDeleteConfirmDialog: true },
      priorityRules: { required: true, minLength: [(args) => {return
args['value'].length <= 8;}, 'Value should be within 8 letters'] }
    };
    },
    provide: {
    treegrid: [ Edit, Toolbar ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/editing/default-cs15" %}

Persisting data in server in Vue Treegrid component

Edited data can be persisted in the database using the RESTful web services.

All the CRUD operations in the treegrid are done through DataManager. The DataManager has an option to bind all the CRUD related data in server-side.

For your information, the ODataAdaptor persists data in the server as per OData protocol.

In the following section, we have explained how to perform CRUD operation in server-side using the UrlAdaptor and RemoteSave Adaptor.

URL adaptor

You can use the UrlAdaptor of DataManager when binding data source from remote data. In the initial load of treegrid, data are fetched from remote data and bound to the treegrid using url property of DataManager. You can map The CRUD operation in treegrid can be mapped to server-side Controller actions using the properties insertUrl, removeUrl, updateUrl and batchUrl.

The following code example describes the above behavior.

`

<template>

<ejs-treegrid :dataSource='data' idMapping='TaskID' parentIdMapping='parentID' :treeColumnIndex='1' height='270px' :editSettings='editSettings' :toolbar='toolbar'>

<e-columns>

Copyright © 2001 -2024 Syncfusion Inc.

```
<e-column field='TaskID'  :isPrimaryKey='true'  headerText='Task ID'  width=90 textAlign='Right'></e-column>

<e-column field='TaskName' headerText='Task Name' width=180></e-column>

<e-column field='StartDate' headerText='Start Date' editType= 'datepickeredit' type= 'date' format= 'yMd' textAlign='Right' width=170></e-column>

<e-column field='EndDate' headerText='End Date' editType= 'datepickeredit' type= 'date' format= 'yMd' textAlign='Right' width=170></e-column>

<e-column field='Priority' headerText='Priority' width=110></e-column>

</ejs-treegrid>

</div>

</template>

<script>

import Vue from "vue";

import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-treegrid";

import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';

Vue.use(TreeGridPlugin);

export default {

data() {

return {

data: new DataManager({

url: "Home/DataSource",

updateUrl: "Home/Update",

insertUrl: "Home/Insert",

removeUrl: "Home/Delete",

batchUrl: "Home/Remove",

adaptor: new UrlAdaptor

}),

toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],

editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Row', newRowPosition: 'Below' }

};

},

provide: {

treegrid: [ Edit, Toolbar ]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
}
</script>
```

Also, when using the UrlAdaptor, you need to return the data as JSON from the controller action and the JSON object must contain a property as result with dataSource as its value and one more property count with the dataSource total records count as its value.

The following code example describes the above behavior.

```ts
public ActionResult DataSource(DataManager dm)
{
var DataSource = TreeData.GetTree();
DataOperations operation = new DataOperations();
if (dm.Where != null && dm.Where.Count > 0)
{
DataSource = operation.PerformFiltering(DataSource, dm.Where, "and");   //perform filtering  and maintain child records on Expand/Collapse operation
}
var count = DataSource.ToList<TreeData>().Count();
if (dm.Skip != 0)
{
DataSource = operation.PerformSkip(DataSource, dm.Skip);   //Paging
}
if (dm.Take != 0)
{
DataSource = operation.PerformTake(DataSource, dm.Take);
}
return dm.RequiresCounts ? Json(new { result = DataSource, count = count }) : Json(DataSource);
}
```

Insert record

Using the insertUrl property, you can specify the controller action mapping URL to perform insert operation on the server-side.

The following code example describes the above behavior and also we have inserted new record based on the newRowPosition TreeGrid editSettings as "Below".

```ts
public void Insert(TreeGridData value, int relationalKey)
{
var i = 0;
for (; i < TreeData.tree.Count; i++)
{
if (TreeData.tree[i].TaskID == relationalKey)
{
break;
}
}
i += FindChildRecords(relationalKey); // Inserted new record when newRowPosition API is in "Below".
TreeData.tree.Insert(i + 1, value);
}
public int FindChildRecords(int id)
{
var count = 0;
for (var i = 0; i < TreeData.tree.Count; i++)
{
if (TreeData.tree[i].ParentItem == id)
{
count++;
count += FindChildRecords(TreeData.tree[i].TaskID);
}
}
return count;
}
```

The newly added record details are bound to the value parameter and relationalKey contains primaryKey value of an selected record helps to find out the position of newly added record. Please refer to the following screenshot.

![Insert](images/insert.PNG)

Copyright © 2001 -2024 Syncfusion Inc.

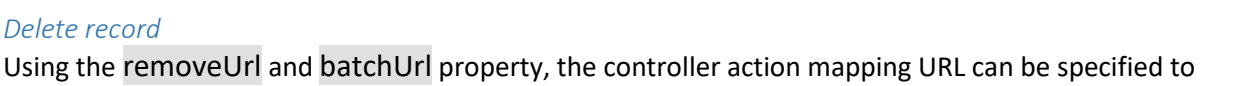
Update record

Using the updateUrl property, the controller action mapping URL can be specified to perform save/update operation on the server-side.

The following code example describes the previous behavior.

```ts
public ActionResult Update(TreeGridData value)
{
var val = TreeData.tree.Where(ds => ds.TaskID == value.TaskID).FirstOrDefault();
val.TaskName = value.TaskName;
val.StartDate = value.StartDate;
val.Duration = value.Duration;
val.Priority = value.Priority;
val.Progress = value.Progress;
return Json(value);
}
```

The updated record details are bound to the value parameter. Please refer to the following screenshot.



Delete record

Using the removeUrl and batchUrl property, the controller action mapping URL can be specified to perform delete operation on the server-side.

The following code example describes the previous behavior.

```ts
public ActionResult Delete(int key)
{
TreeData.tree.Remove(TreeData.tree.Where(ds => ds.TaskID == key).FirstOrDefault());
}
// Remove method (batchUrl) will be triggered when we delete parent record.
public ActionResult Remove(List<TreeGridData> changed, List<TreeGridData> added,
List<TreeGridData> deleted)
{
for (var i = 0; i < deleted.Count; i++)
{
TreeData.tree.Remove(TreeData.tree.Where(ds => ds.TaskID == deleted[i].TaskID).FirstOrDefault());
}
```

Copyright © 2001 -2024 Syncfusion Inc.

}

`

The deleted record primary key value is bound to the key parameter. Please refer to the following screenshot.

![Delete](images/remove.PNG)

While delete parent record, the parent and child records is bound to the deleted parameter. Please refer to the following screenshot.

![Remove](images/delete.PNG)

Remote save adaptor

You may need to perform all Tree Grid Actions in client-side except the CRUD operations, that should be interacted with server-side to persist data. It can be achieved in TreeGrid by using **RemoteSaveAdaptor**.

Datasource must be set to **json** property and set **RemoteSaveAdaptor** to the **adaptor** property. CRUD operations can be mapped to server-side using **updateUrl**, **insertUrl**, **removeUrl** properties.

You can use the following code example to use **RemoteSaveAdaptor** in TreeGrid.

`

<template>

<ejs-treegrid :dataSource='data' idMapping='TaskID' parentIdMapping='parentID' :treeColumnIndex='1' height='270px' :editSettings='editSettings' :toolbar='toolbar'>

<e-columns>

<e-column field='TaskID' :isPrimaryKey='true' headerText='Task ID' width=90 textAlign='Right'></e-column>

<e-column field='TaskName' headerText='Task Name' width=180></e-column>

<e-column field='StartDate' headerText='Start Date' editType= 'datepickeredit' type= 'date' format= 'yMd' textAlign='Right' width=170></e-column>

<e-column field='EndDate' headerText='End Date' editType= 'datepickeredit' type= 'date' format= 'yMd' textAlign='Right' width=170></e-column>

<e-column field='Priority' headerText='Priority' width=110></e-column>

</ejs-treegrid>

</template>

<script>

import Vue from "vue";

import { TreeGridPlugin, Edit, Toolbar } from "@syncfusion/ej2-vue-treegrid";

import { DataManager, RemoteSaveAdaptor } from '@syncfusion/ej2-data';

import { projectData } from "./datasource.js";

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(TreeGridPlugin);
export default {
data() {
return {
data: new DataManager({
data: projectData
updateUrl: "Home/Update",
insertUrl: "Home/Insert",
removeUrl: "Home/Delete",
batchUrl: "Home/Remove",
adaptor: new RemoteSaveAdaptor
}),
toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Row', newRowPosition: 'Below' }
};
},
provide: {
treegrid: [ Edit, Toolbar ]
}
}
</script>
`
```

The following code example describes the CRUD operations handled at server-side.

```ts
public void Insert(TreeData value, int relationalKey)
{
var i = 0;
for (; i < TreeData.tree.Count; i++)
{
if (TreeData.tree[i].TaskID == relationalKey)
{
break;
```

```
}
}
i += FindChildRecords(relationalKey); // Inserted new record when newRowPosition API is in "Below".
TreeData.tree.Insert(i + 1, value);
}
public ActionResult Update(TreeData value)
{
var val = TreeData.tree.Where(ds => ds.TaskID == value.TaskID).FirstOrDefault();
val.TaskName = value.TaskName;
val.StartDate = value.StartDate;
val.Duration = value.Duration;
val.Priority = value.Priority;
val.Progress = value.Progress;
return Json(value);
}
public ActionResult Delete(int key)
{
TreeData.tree.Remove(TreeData.tree.Where(ds => ds.TaskID == key).FirstOrDefault());;
}
// Remove method (batchUrl) will be triggered when we delete parent record.
public ActionResult Remove(List<TreeGridData> changed, List<TreeGridData> added,
List<TreeGridData> deleted)
{
for (var i = 0; i < deleted.Count; i++)
{
TreeData.tree.Remove(TreeData.tree.Where(ds => ds.TaskID == deleted[i].TaskID).FirstOrDefault());
}
}
`
```

Sorting in Vue Treegrid component

Sorting enables you to sort data in the Ascending or Descending order.

To sort a column, click the column header.

Copyright © 2001 -2024 Syncfusion Inc.

To sort multiple columns, press and hold the CTRL key and click the column header. You can clear sorting of any one of the multi-sorted columns by pressing and holding the SHIFT key and clicking the specific column header.

To enable sorting in the TreeGrid, set the allowSorting to true. Sorting options can be configured through the sortSettings.

To sort, inject the Sort module in the treegrid.

APP.VUE

```
<template>
    <div id="app">
          <ejs-treegrid  :dataSource='data' :allowSorting='true'
height='315px' childMapping='subtasks' :treeColumnIndex='0'>
            <e-columns>
              <e-column field='Category' headerText='Category'
width='140'></e-column>
                <e-column field='orderName' headerText='Order Name'
width='200'></e-column>
                <e-column field='orderDate' headerText='Order Date'
width='150' format="yMd" textAlign='Right'></e-column>
                <e-column field='units' headerText='Units' width='90'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
        </div>
      </template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Sort } from "@syncfusion/ej2-vue-treegrid";
import { sortData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: sortData
    };
  },
  provide: {
      treegrid: [ Sort ]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/sorting/default-cs1" %}

* TreeGrid columns are sorted in the Ascending order. If you click the already sorted column, the sort direction toggles.

* You can apply and clear sorting by invoking sortByColumn and

clearSorting methods.

* To disable sorting for a particular column, set the columns.allowSorting to false.

Copyright © 2001 -2024 Syncfusion Inc.

Initial sort

To sort at initial rendering, set the field and direction in the sortSettings.columns.

APP.VUE

```
<template>
    <div id="app">
          <ejs-treegrid  :dataSource='data' :allowSorting='true'
:sortSettings='sortSettings' childMapping='subtasks' :treeColumnIndex='1'>
            <e-columns>
              <e-column field='Category' headerText='Category'
width='140'></e-column>
                <e-column field='orderName' headerText='Order Name'
width='200'></e-column>
                <e-column field='orderDate' headerText='Order Date'
width='120' format="yMd" textAlign='Right'></e-column>
                <e-column field='units' headerText='Units' width='90'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
    </div>
      </template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Sort } from "@syncfusion/ej2-vue-treegrid";
import { sortData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: sortData,
      sortSettings: { columns: [{ field: 'Category', direction: 'Ascending'
}, { field: 'orderName', direction: 'Ascending' }] }
    };
    },
  provide: {
      treegrid: [ Sort ]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/sorting/default-cs2" %}

Sorting events

During the sort action, the treegrid component triggers two events. The actionBegin event triggers before the sort action starts, and the actionComplete event triggers after the sort action is completed. Using these events you can perform the needed actions.

APP.VUE

```
<template>
    <div id="app">
          <ejs-treegrid :dataSource='data' :allowSorting='true'
height='315px' childMapping='subtasks' :treeColumnIndex='1'
:actionComplete='actionHandler' :actionBegin='actionHandler'>
            <e-columns>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-column field='Category' headerText='Category'
width='140'></e-column>
                <e-column field='orderName' headerText='Order Name'
width='200'></e-column>
                <e-column field='orderDate' headerText='Order Date'
width='120' format="yMd" textAlign='Right'></e-column>
                <e-column field='units' headerText='Units' width='90'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
    </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Sort } from "@syncfusion/ej2-vue-treegrid";
import { sortData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: sortData
    };
  },
  provide: {
      treegrid: [ Sort ]
    },
    methods: {
    actionHandler: function(args) {
        alert(args.requestType + ' ' + args.type); // custom Action
    }
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/sorting/default-cs3" %}

The args.requestType is the current action name. For example, in sorting the args.requestType value is 'sorting'.

Touch interaction

When you tap the treegrid header on touchscreen devices, the selected column header is sorted. A popup is displayed for multi-column sorting. To sort multiple columns, tap the popup and then tap the desired treegrid headers.

The following screenshot shows treegrid touch sorting.

<img src="./images/touch-sorting.jpg" alt="Touch interaction image" style="width:320px;height:620px">

Copyright © 2001 -2024 Syncfusion Inc.

Filtering

Filtering in Vue Treegrid component

Filtering allows you to view specific or related records based on filter criteria. To enable filtering in the TreeGrid, set the allowFiltering to true. Filtering options can be configured through filterSettings.

To use filter, inject the Link to the Video module in the treegrid.

To get start quickly with filtering options, you can check on this video:

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :allowFiltering='true' height='275px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
    };
  },
  provide: {
      treegrid: [ Filter ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/filtering/default-cs6" %}

* You can apply and clear filtering by using filterByColumn and clearFiltering methods.

* To disable filtering for a particular column, set

columns.allowFiltering to false.

Filter hierarchy modes

TreeGrid provides support for a set of filtering modes with filterSettings.filterHierarchyMode property.

The below are the type of filter mode available in TreeGrid.

Copyright © 2001 -2024 Syncfusion Inc.

- **Parent** : This is the default filter hierarchy mode in TreeGrid. The filtered records are displayed with its parent records, if the filtered records not have any parent record then the filtered records only displayed.
- **Child** : The filtered records are displayed with its child record, if the filtered records not have any child record then the filtered records only displayed.
- **Both** : The filtered records are displayed with its both parent and child record, if the filtered records not have any parent and child record then the filtered records only displayed.
- **None** : The filtered records are only displayed.

APP.VUE

```
<template>
<div id="app">
 <div style="padding-top: 7px; display: inline-block">Hierarchy Mode</div>
  <div style="display: inline-block; width: 150px">
<ejs-dropdownlist id='mode' :dataSource='ddldata'  :fields='fields'
:value='value' :popupHeight='height' :change='change'></ejs-dropdownlist>
</div>
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1'  :allowFiltering='true'
height='225px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='75'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter, TreeGridComponent  } from "@syncfusion/ej2-
vue-treegrid";
import { DropDownListPlugin, ChangeEventArgs } from '@syncfusion/ej2-vue-
dropdowns';
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
Vue.use(DropDownListPlugin);
export default {
  data () {
    return {
      data: sampleData,
      height: '220px',
      ddldata : [{ id: 'Parent', mode: 'Parent' },
      { id: 'Child', mode: 'Child' },
      { id: 'Both', mode: 'Both' },
      { id: 'None', mode: 'None' }
    ],
    fields: { text: 'mode', value: 'id' },
    value: 'Parent'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    };
  },
    provide: {
      treegrid: [ Filter ]
    },
    methods: {
        change: function (e: ChangeEventArgs)  {
        let mode: any = e.value;
        this.$refs.treegrid.ej2Instances.filterSettings.hierarchyMode =
mode;
        this.$refs.treegrid.clearFiltering();
    }
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/filtering/default-cs7" %}

Initial filter

To apply the filter at initial rendering, set the filter predicate object in

filterSettings.columns.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1'  :allowFiltering='true'
:filterSettings='filterOptions' height='275px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='75'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: sampleData,
      filterOptions: {
            columns: [{ field: 'taskName', matchCase: false, operator:
'startswith', predicate: 'and', value: 'plan' },
            { field: 'duration', matchCase: false, operator: 'equal',
predicate: 'and', value: 5 }]
```

```
      }
    };
  },
    provide: {
      treegrid: [ Filter ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/filtering/default-cs8" %}

Filter operators

The filter operator for a column can be defined in the filterSettings.columns.operator property.

The available operators and its supported data types are:

Operator |Description |Supported Types

startswith |Checks whether the value begins with the specified value. |String

endswith |Checks whether the value ends with the specified value. |String

contains |Checks whether the value contains the specified value. |String

equal |Checks whether the value is equal to the specified value. |String | Number | Boolean | Date

notequal |Checks for values not equal to the specified value. |String | Number | Boolean | Date

greaterthan |Checks whether the value is greater than the specified value. |Number | Date

greaterthanorequal|Checks whether a value is greater than or equal to the specified value. |Number | Date

lessthan |Checks whether the value is less than the specified value. |Number | Date

lessthanorequal |Checks whether the value is less than or equal to the specified value. |Number | Date

By default, the filterSettings.columns.operator value is equal.

Change default filter operator

You can change the default filter operator by extending filterModule.filterOperatorsproperty in dataBound event. In the following sample, we have changed the default operator for string typed columns as contains from startsWith.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid':dataSource='data'
childMapping='subtasks' :treeColumnIndex='1'  :allowFiltering='true'
height='275px' :dataBound='dataBound'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='75'
textAlign='Right'></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
    };
  },
   provide: {
    treegrid: [Filter]
  },
  methods: {
    dataBound(args){

this.$refs.treegrid.ej2Instances.grid.filterModule.filterOperators.startsWit
h="contains"
   }
    }
  }
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/filtering/default-cs9" %}

Diacritics

By default, treegrid ignores diacritic characters while filtering. To include diacritic characters, set the filterSettings.ignoreAccent as true.

In the following sample, type **aero** in Name column to filter diacritic characters.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='Children'
:treeColumnIndex='0'  :allowFiltering='true' height='275px'
:filterSettings='filterSettings'>
            <e-columns>
                <e-column field='EmpID' headerText='Employee ID' width='90'
textAlign='Right'></e-column>
                <e-column field='Name' headerText='Name' width='110'></e-
column>
                <e-column field='DOB' headerText='DOB' width='90'
format="yMd" textAlign='Right'></e-column>
```

```
                <e-column field='Country' width='65'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter } from "@syncfusion/ej2-vue-treegrid";
import { diacritics } from "./datasource.js";
Vue.use(TreeGridPlugin),
export default {
  data ()  {
    return {
      data: diacritics,
       filterSettings: {
        ignoreAccent: true
       }
    };
    },
  provide: {
    treegrid: [Filter]
  }
  }
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/filtering/default-cs10" %}

Filter bar in Vue Treegrid component

By setting the allowFiltering to true, the filter bar row will render next to the header, which allows you to filter data. You can filter the records with different expressions depending upon the column type.

Filter bar expressions:

You can enter the following filter expressions (operators) manually in the filter bar.

Expression |Example |Description |Column Type

= |=value |equal |Number

!= |!=value |notequal |Number

|>value |greaterthan |Number

< |<value |lessthan |Number

= |>=value |greaterthanorequal |Number

<=|<=value|lessthanorequal |Number

/value |startswith |String

% |%value |endswith |String

N/A |N/A | Equal operator will always be used for date filter. |Date

N/A |N/A |Equal operator will always be used for Boolean filter. |Boolean

APP.VUE

```
<template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1'  :allowFiltering='true' height='275px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='75'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
    };
  },
   provide: {
    treegrid: [Filter]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/filtering/default-cs3" %}

Filter menu in Vue Treegrid component

You can enable filter menu by setting the filterSettings.type as Menu. The filter menu UI will be rendered based on its column type, which allows you to filter data.

You can filter the records with different operators.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1'  :allowFiltering='true' height='275px'
:filterSettings='filterSettings'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='75'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='120'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
            </e-columns>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      data: sampleData,
      filterSettings: {
           type: 'Menu'
      }
    };
  },
  provide: {
      treegrid: [ Filter ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/filtering/default-cs4" %}

* allowFiltering must be set as true to enable filter menu.

* Setting columns.allowFiltering as false will prevent filter menu rendering for a particular column.

Enable different filter dialog for a column

You can use both Menu and Excel filter in a same TreeGrid. To do so, set the

column.filter.type as Menu or Excel.

In the following sample menu filter is enabled by default and Excel filter is enabled for the Task Name column using the

column.filter.type.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :allowFiltering='true' height='273px'
:filterSettings='filterOptions'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160' :filter='filter'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      filterOptions: {
        type: 'Menu'
      },
      filter: {
        type : 'Excel';
      }
    };
  },
  provide: {
      treegrid: [ Filter ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/filtering/default-cs5" %}

Excel like filter in Vue Treegrid component

You can enable Excel like filter by defining. filterSettings.type as Excel.The excel menu contains an option such as Sorting, Clear filter, Sub menu for advanced filtering.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :allowFiltering='true' height='273px'
:filterSettings='filterOptions'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160' ></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data ()  {
    return {
      data: sampleData,
      filterOptions: {
        type: 'Excel'
      }
    };
  },
  provide: {
      treegrid: [ Filter ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/filtering/default-cs1" %}

Change default excel filter operator

You can change the default excel-filter operator by changing the column operator as `contains` from `startsWith` in the actionBegin event.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :allowFiltering='true' height='273px'
:filterSettings='filterOptions' :actionBegin = 'actionBegin'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160' ></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      filterOptions: {
        type: 'Excel'
      }
    };
  },
  provide: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      treegrid: [ Filter ]
    },
     methods: {
    actionBegin(args) {
       if(args.requestType === 'filtersearchbegin' && args.column.type ===
"string")
      {
        args.operator = 'contains';
      }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/filtering/default-cs2" %}

Searching in Vue Treegrid component

You can search records in a TreeGrid, by using the search method with search key as a parameter. This also provides an option to integrate search text box in treegrid's toolbar by adding search item to the toolbar.

To search records, inject the Filter module in the treegrid.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :toolbar='toolbarOptions' height='270px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['Search']
    };
  },
  provide: {
```

```
      treegrid: [ Filter, Toolbar ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/searching/default-cs1" %}

Initial search

To apply search at initial rendering, set the fields, operator, key, and ignoreCase in the searchSettings.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :toolbar='toolbarOptions' height='270px'
:searchSettings='searchOptions' >
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
        data: sampleData,
        toolbarOptions: ['Search'],
        searchOptions: { fields: ['taskName'], operator: 'contains', key:
'plan', ignoreCase: true }
    };
  },
  provide: {
      treegrid: [ Filter, Toolbar ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/searching/default-cs2" %}

By default, treegrid searches all the bound column values. To customize this behavior define the searchSettings.fields property.

Copyright © 2001 -2024 Syncfusion Inc.

Search operators

The search operator can be defined in the searchSettings.operator property to configure specific searching.

The following operators are supported in searching:

Operator |Description

startsWith |Checks whether a value begins with the specified value.

endsWith |Checks whether a value ends with the specified value.

contains |Checks whether a value contains the specified value.

equal |Checks whether a value is equal to the specified value.

notEqual |Checks for values not equal to the specified value.

By default, the searchSettings.operator value is contains.

Search by external button

To search treegrid records from an external button, invoke the search method.

APP.VUE

```
<template>
<div id="app">
<div class="e-float-input" style="width: 200px; display: inline-block;">
                <input type="text" class="searchtext"/>
                <span class="e-float-line"></span>
                <label class="e-float-text">Search text</label>
            </div>
        <ejs-button id='search' @click.native='search'>Search</ejs-button>
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1' height='220px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter } from "@syncfusion/ej2-vue-treegrid";
import { ButtonPlugin } from '@syncfusion/ej2-vue-buttons';
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
Vue.use(ButtonPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    };
  },
    provide: {
      treegrid: [ Filter ]
    },
    methods:{
        search: function() {
        let searchText =
document.getElementsByClassName('searchtext')[0].value;
        this.$refs.treegrid.search(searchText);
    }
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/searching/default-cs3" %}

Search specific columns

By default, treegrid searches all visible columns. You can search specific columns by defining the specific column's field names in the searchSettings.fields property.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :toolbar='toolbarOptions' height='270px'
:searchSettings='searchOptions'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Filter, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['Search'],
      searchOptions: {fields: ['taskName', 'duration']}
    };
  },
  provide: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      treegrid: [ Filter, Toolbar ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/searching/default-cs4" %}

Paging in Vue Treegrid component

Paging provides an option to display TreeGrid data in page segments. To enable paging, set the allowPaging to true. When paging is enabled, pager component renders at the bottom of the treegrid. Paging options can be configured through the pageSettings.

To use paging, inject the Page module in the grid.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :allowPaging='true' :pageSettings='pageSettings'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      pageSettings: { pageSize: 7 }
    };
  },
  provide: {
      treegrid: [ Page ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/paging/default-cs1" %}

You can achieve better performance by using treegrid paging to fetch only a pre-defined number of records from the data source.

Copyright © 2001 -2024 Syncfusion Inc.

Page Size Mode

Two behaviors are available in TreeGrid paging to display certain number of records in a current page. Following are the two types of pageSizeMode.

- **All** : This is the default mode. The number of records in a page is based on pageSize property.
- **Root** : The number of root nodes or the 0th level records to be displayed per page is based on pageSize property.

With pageSizeMode property as Root, only the root level or the 0th level records are considered in records count.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :allowPaging='true' :pageSettings='pageSettings'
height='265px' >
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      pageSettings: { pageSize: 2, pageSizeMode: 'Root'}
    };
  },
  provide: {
      treegrid: [ Page ]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/paging/default-cs2" %}

Pager with Page Size Dropdown

The pager Dropdown allows you to change the number of records in the TreeGrid dynamically. It can be enabled by defining the pageSettings.pageSizes property as true.

Copyright © 2001 -2024 Syncfusion Inc.

`

pageSettings: {pageSize: 7, pageSizes: true},

`

<!-- markdownlint-disable MD033 -->

<img src="./images/pagesizes.png" alt="Page size dropdown">

<!-- markdownlint-enable MD033 -->

How to render Pager at the Top of the TreeGrid

By default, Pager will be rendered at the bottom of the TreeGrid. You can also render the Pager at the top of the TreeGrid by using the dataBound event.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :dataBound='ondataBound'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, TreeGridComponent,  Page } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
let initialGridLoad = true;
export default {
  data ()  {
    return {
      data: sampleData,
      pageSettings: { pageSize: 7, pageSizes: true },
      initialGridLoad : true
      };
  },
  provide: {
      treegrid: [ Page ]
    },
     methods: {
    ondataBound: function() {
      if (this.initialGridLoad) {
          this.initialGridLoad = false;
```

```
            let pager = document.getElementsByClassName('e-gridpager e-control
e-pager e-lib');
            let topElement;
           if ( this.$refs.treegrid.toolbar) {
               topElement = document.getElementsByClassName('e-toolbar');
          } else {
            topElement = document.getElementsByClassName('e-gridheader e-lib
e-droppable');
          }
            topElement[0].before(pager[0]);
        }
    }
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/paging/default-cs3" %}

During the paging action, the pager component triggers the below three events.

* The created event triggers when Pager is created.

* The click event triggers when the numeric items in the pager is clicked.

* The dropDownChanged event triggers when pageSize DropDownList value is selected.

Scrolling in Vue Treegrid component

The scrollbar will be displayed in the treegrid when content exceeds the element width or height. The vertical and horizontal scrollbars will be displayed based on the following criteria:

- The vertical scrollbar appears when the total height of rows present in the treegrid exceeds its element height.
- The horizontal scrollbar appears when the sum of columns width exceeds the treegrid element width.
- The height and width are used to set the treegrid height and width, respectively.

The default value for height and width is auto.

Set width and height

To specify the width and height of the scroller in the pixel, set the pixel value to a number.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :height='315' :width='400'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='120' format="yMd" textAlign='Right'></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-column field='duration' headerText='Duration' width='110'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData
    };
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/scroll/default-cs1" %}

Responsive with parent container

Specify the width and height as 100% to make the treegrid element fill its parent container.

Setting the height to 100% requires the treegrid parent element to have explicit height.

APP.VUE

```
<template>
<div id="app">
    <p class="e-text"> The parent container can be resizable by dragging the
bottom-right corner.</p>
      <div id='container' class='e-resizable'>
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
height='100%' width= '100%' :treeColumnIndex='1' >
            <e-columns>
              <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
              <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
              <e-column field='startDate' headerText='Start Date'
width='120' format="yMd" textAlign='Right'></e-column>
              <e-column field='duration' headerText='Duration' width='110'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
      </div>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  data ()  {
    return {
      data: sampleData
    };
  }
}
</script>
<style>
  .e-resizable {
      resize: both;
      overflow: auto;
      border: 1px solid red;
      padding: 10px;
      height: 300px;
      min-height: 250px;
      min-width: 250px;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/scroll/default-cs2" %}

Scroll to selected row

You can scroll the treegrid content to the selected row position by using the rowSelected event.

APP.VUE

```
<template>
<div id="app">
    <ejs-numerictextbox ref='numeric' format='N' min='0' placeholder='Enter
index to select a row' width=200 :showSpinButton='false'
:change='onchange'></ejs-numerictextbox>
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' height='280' width='100%' :treeColumnIndex='1'
:rowSelected='rowSelected'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='120' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='110'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { NumericTextBoxPlugin } from "@syncfusion/ej2-vue-inputs";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
Vue.use(NumericTextBoxPlugin);
export default {
  data ()  {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    return {
      data: sampleData
    };
  },
  methods: {
    onchange: function(){
      this.$refs.treegrid.selectRow(parseInt(this.$refs.numeric.getText(),
10));
    }
    rowSelected: function (args) {
        let rowHeight =
this.$refs.treegrid.getRows()[this.$refs.treegrid.getSelectedRowIndexes()[0]
].scrollHeight;
        this.$refs.treegrid.getContent().children[0].scrollTop = rowHeight *
this.$refs.treegrid.getSelectedRowIndexes()[0];
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/scroll/selectrow-cs1" %}

Frozen in Vue Treegrid component

Frozen rows and columns

Frozen rows and columns provides an option to make rows and columns always visible in the top and left side of the tree grid while scrolling.

To use frozen rows and columns support, inject the `Freeze` module in the `provide` section.

In this demo, the frozenColumns is set as '2' and the frozenRows

is set as '3'. Hence, the left two columns and top three rows are frozen.

APP.VUE

```
<template>
    <div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height=310 :frozenColumns='2' :frozenRows='3'
:allowSelection='false'>
            <e-columns>
               <e-column field='taskID' headerText='Task ID' width='110'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='230'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='120' format="yMd" textAlign='Right'></e-column>
                <e-column field='endDate' headerText='End Date' width='120'
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='110'
textAlign='Right'></e-column>
                <e-column field='progress' headerText='Progress' width='120'
textAlign='Right'></e-column>
                <e-column field='priority' headerText='Priority'
width='120'></e-column>
```

```
                <e-column field='approved' headerText='Approved' width='110'
textAlign='Left'></e-column>
            </e-columns>
        </ejs-treegrid>
    </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Freeze } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData
    };
  },
  provide: {
    treegrid: [Freeze]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/scroll/frozencolumn-cs1" %}

Freeze particular columns

You can use isFrozen property to freeze selected columns in tree grid.

In this demo, the columns with field name taskName and startDate is frozen using

the isFrozen property.

APP.VUE

```
<template>
    <div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks' height=310
:allowSelection='false'>
            <e-columns>
               <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='230' isFrozen='true'></e-column>
                <e-column field='startDate' headerText='Start Date'
isFrozen='true' width='120' format="yMd" textAlign='Right'></e-column>
                <e-column field='endDate' headerText='End Date' width='120'
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='110'
textAlign='Right'></e-column>
                <e-column field='progress' headerText='Progress' width='120'
textAlign='Right'></e-column>
                <e-column field='priority' headerText='Priority'
width='120'></e-column>
                <e-column field='approved' headerText='Approved' width='110'
textAlign='Left'></e-column>
            </e-columns>
        </ejs-treegrid>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Freeze } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData
    };
  },
  provide: {
    treegrid: [Freeze]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/scroll/isfrozen-cs1" %}

Freeze direction

You can freeze the tree grid columns on the left or right side by using the column.freeze property and the remaining columns will be movable. The tree grid will automatically move the columns to the left or right position based on the column.freeze value.

Types of the column.freeze directions:

- **Left**: Allows you to freeze the columns at the left.
- **Right**: Allows you to freeze the columns at the right.

In this demo, the **Task Name** column is frozen at the left and the **Priority** column is frozen at the right side of the content table.

APP.VUE

```
<template>
    <div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks' height=310
:treeColumnIndex='1'  :allowSelection='false'>
            <e-columns>
               <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='230' freeze='Left'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='120' format="yMd" textAlign='Right'></e-column>
                <e-column field='endDate' headerText='End Date' width='120'
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='110'
textAlign='Right'></e-column>
                <e-column field='progress' headerText='Progress' width='120'
textAlign='Right'></e-column>
                <e-column field='priority' headerText='Priority' width='120'
freeze='Right'></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-column field='approved' headerText='Approved' width='110'
textAlign='Left'></e-column>
            </e-columns>
        </ejs-treegrid>
    </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Freeze } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData
    };
  },
  provide: {
    treegrid: [Freeze]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/scroll/freeze-direction-cs1" %}

* Freeze Direction is not compatible with the isFrozen and frozenColumns properties.

Limitations of frozen tree grid

The following features are not supported in frozen rows and columns:

- Row Template
- Detail Template
- Cell Editing

Freeze Direction feature has the below limitations, along with the above mentioned limitations.

- Infinite scroll cache mode
- Freeze direction in the stacked header is not compatible with column reordering.

Virtual scroll in Vue Treegrid component

TreeGrid allows you to load large amount of data without performance degradation.

To use virtualization, you need to inject VirtualScroll module in the provide section.

Row virtualization

Row virtualization allows you to load and render rows only in the content viewport. It is an alternative way of paging in which the rows will be appended while scrolling vertically. To setup the row virtualization, you need to define enableVirtualization as true and content height by height property.

The number of records displayed in the TreeGrid is determined implicitly by height of the content area and a buffer records will be maintained in the TreeGrid content in addition to the original set of rows.

Expand and Collapse state of any child record will be persisted.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
<div id="app">
<ejs-treegrid ref='treegrid' :dataSource="virtualData"
:enableVirtualization='true' :treeColumnIndex='1' childMapping='Crew'
height=317>
    <e-columns>
        <e-column field='TaskID' headerText='Player Jersey' width='120'
textAlign='Right'></e-column>
        <e-column field='FIELD1' headerText='Player Name' width='120'></e-
column>
        <e-column field='FIELD2' headerText='Year' width='100'
textAlign='Right'></e-column>
        <e-column field='FIELD3' headerText='Stint' width='120'
textAlign='Right'></e-column>
        <e-column field='FIELD4' headerText='TMID' width='120'
textAlign='Right'></e-column>
        <e-column field='FIELD5' headerText='LGID' width='120'
textAlign='Right'></e-column>
    </e-columns>
</ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, VirtualScroll } from "@syncfusion/ej2-vue-
treegrid";
import { dataSource, virtualData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      virtualData: dataSource(),
    };
  },
  provide: {
      treegrid: [VirtualScroll]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/scroll/virtualscroll-cs4" %}

Column virtualization

Column virtualization allows you to virtualize columns. It will render column only in the current view port and all other columns are rendered on demand during horizontal scrolling.

To setup the column virtualization, set the enableVirtualization and enableColumnVirtualization properties as true.

APP.VUE

```
<template>
<div id="app">
```

```
<ejs-treegrid ref='treegrid' :dataSource="virtualData"
:enableVirtualization='true' :enableColumnVirtualization='true'
:treeColumnIndex='1' childMapping='Crew' height=317>
    <e-columns>
        <e-column field='TaskID' headerText='Player Jersey' width='120'
textAlign='Right'></e-column>
        <e-column field='FIELD1' headerText='Player Name' width='120'></e-
column>
        <e-column field='FIELD2' headerText='Year' width='100'
textAlign='Right'></e-column>
        <e-column field='FIELD3' headerText='Stint' width='120'
textAlign='Right'></e-column>
        <e-column field='FIELD4' headerText='TMID' width='120'
textAlign='Right'></e-column>
        <e-column field='FIELD5' headerText='LGID' width='120'
textAlign='Right'></e-column>
        <e-column field='FIELD5' headerText='LGID' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD6' headerText='GP' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD7' headerText='GS' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD8' headerText='Minutes' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD9' headerText='Points' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD10' headerText='oRebounds' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD11' headerText='dRebounds' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD12' headerText='Rebounds' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD13' headerText='Assists' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD14' headerText='Steals' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD15' headerText='Blocks' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD16' headerText='Turnovers' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD17' headerText='PF' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD18' headerText='fgAttempted' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD19' headerText='ftAttempted' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD20' headerText='ThreeAttempted' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD21' headerText='ThreeMade' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD22' headerText='PostGP' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD23' headerText='ftMade' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD24' headerText='fgMade' width= 120
textAlign='Right'></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        <e-column field='FIELD25' headerText='ffmade' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD26' headerText='PostGS' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD27' headerText='PostMinutes' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD28' headerText='PostPoints' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD29' headerText='PostoRebounds' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD30' headerText='PostdRebounds' width= 120
textAlign='Right'></e-column>
    </e-columns>
</ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, VirtualScroll } from "@syncfusion/ej2-vue-
treegrid";
import { dataSource, virtualData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data () {
    return {
      virtualData: dataSource(),
    };
  },
  provide: {
      treegrid: [VirtualScroll]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/scroll/virtualscroll-cs5" %}

Column's width is required for column virtualization. If column's width is not defined then tree grid will consider its value as 200px.

Limitations for virtualization

- Due to the element height limitation in browsers, the maximum number of records loaded by the treegrid is limited by the browser capability.
- Cell selection will not be persisted in row.
- Virtual scrolling is not compatible with detail template,clipboard functionality, Batch editing and row drag and drop features.
- The page size provided must be two times larger than the number of visible rows in the TreeGrid. If the page size is failed to meet this condition then the size will be determined by TreeGrid.
- The virtual height of the treegrid content is calculated using the row height and total number of records in the data source and hence features which changes row height such as text wrapping are not supported. If you want to increase the row height to accommodate the content then you can specify the row height as below to ensure all the table rows are in same height.

Copyright © 2001 -2024 Syncfusion Inc.

`

.e-treegrid .e-row {

height: 2em;

}

`

- Programmatic selection using the **selectRows** method is not supported in virtual scrolling.
- When virtualization is active in a tree grid, the editCell method is unusable for records outside the currently visible viewport.

Infinite scroll in Vue Treegrid component

Infinite scrolling is used to load a huge amount of data without degrading the TreeGrid performance. This feature works like the lazy loading concept, which means the buffer data is loaded only when the scrollbar reaches the end of the scroller.

To enable Infinite scrolling, set `enableInfiniteScrolling` property as true.

* In this feature, TreeGrid will not make a new data request when you visit the same page again.

APP.VUE

```
<template>
    <div id="app">
        <ejs-treegrid ref='treegrid' :dataSource="virtualData"
:enableInfiniteScrolling='true' :treeColumnIndex='1' childMapping='Crew'
:height=317 :pageSettings='pageSettings'>
            <e-columns>
                <e-column field='TaskID' headerText='Player Jersey'
width='140' textAlign='Right'></e-column>
                <e-column field='FIELD1' headerText='Player Name'
width='140'></e-column>
                <e-column field='FIELD2' headerText='Year' width='105'
textAlign='Right'></e-column>
                <e-column field='FIELD3' headerText='Stint' width='90'
textAlign='Right'></e-column>
                <e-column field='FIELD4' headerText='TMID' width='90'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
    </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, InfiniteScroll, TreeGridComponent, Page } from
"@syncfusion/ej2-vue-treegrid";
import { DataManager, JsonAdaptor } from '@syncfusion/ej2-data';
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { dataSource, virtualData } from './datasource.js';
Vue.use(TreeGridPlugin);
export default {
  data: function() {
    return {
        virtualData: dataSource(),
```

```
        pageSettings: { pageSize: 50 }
    };
  },
  provide: {
      treegrid: [InfiniteScroll, Page]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/scroll/virtualscroll-cs1" %}

InitialBlocks

You can define the initial loading pages count by using infiniteScrollSettings.initialBlocks property. By default, this feature loads three pages in initial rendering.

In the below demo, we have changed this property value to load five page records instead of three.

APP.VUE

```
<template>
    <div id="app">
        <ejs-treegrid ref='treegrid' :dataSource="virtualData"
:enableInfiniteScrolling='true' :treeColumnIndex='1' childMapping='Crew'
:height=317 :pageSettings='pageSettings'
:infiniteScrollSettings='infiniteScrollSettings'>
            <e-columns>
                <e-column field='TaskID' headerText='Player Jersey'
width='140' textAlign='Right'></e-column>
                <e-column field='FIELD1' headerText='Player Name'
width='140'></e-column>
                <e-column field='FIELD2' headerText='Year' width='105'
textAlign='Right'></e-column>
                <e-column field='FIELD3' headerText='Stint' width='90'
textAlign='Right'></e-column>
                <e-column field='FIELD4' headerText='TMID' width='90'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
    </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, InfiniteScroll, TreeGridComponent, Page } from
"@syncfusion/ej2-vue-treegrid";
import { DataManager, JsonAdaptor } from '@syncfusion/ej2-data';
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { dataSource, virtualData } from './datasource.js';
Vue.use(TreeGridPlugin);
export default {
  data: function() {
    return {
        virtualData: dataSource(),
        pageSettings: { pageSize: 50 },
        infiniteScrollSettings : { initialBlocks: 5 }
    };
  },
```

```
  provide: {
      treegrid: [InfiniteScroll, Page]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/scroll/virtualscroll-cs2" %}

Cache Mode

Cache is used to store the loaded rows object in the TreeGrid instance which can be reused for creating the row elements whenever you scroll to already visited page. Also, this mode maintains row elements based on the infiniteScrollSettings.maxBlocks count value, once this limit exceeds then it will remove row elements from DOM for new rows.

To enable the cache mode in Infinite scrolling, set infiniteScrollSettings.enableCache property as true.

APP.VUE

```
<template>
    <div id="app">
        <ejs-treegrid ref='treegrid' :dataSource="virtualData"
:enableInfiniteScrolling='true' :treeColumnIndex='1' childMapping='Crew'
:height=317 :pageSettings='pageSettings'
:infiniteScrollSettings='infiniteScrollSettings'>
            <e-columns>
                <e-column field='TaskID' headerText='Player Jersey'
width='140' textAlign='Right'></e-column>
                <e-column field='FIELD1' headerText='Player Name'
width='140'></e-column>
                <e-column field='FIELD2' headerText='Year' width='105'
textAlign='Right'></e-column>
                <e-column field='FIELD3' headerText='Stint' width='90'
textAlign='Right'></e-column>
                <e-column field='FIELD4' headerText='TMID' width='90'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
    </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, InfiniteScroll, TreeGridComponent, Page } from
"@syncfusion/ej2-vue-treegrid";
import { DataManager, JsonAdaptor } from '@syncfusion/ej2-data';
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { dataSource, virtualData } from './datasource.js';
Vue.use(TreeGridPlugin);
export default {
  data: function() {
    return {
        virtualData: dataSource(),
        pageSettings: { pageSize: 50 },
        infiniteScrollSettings: { enableCache: true}
    };
  },
  provide: {
```

```
        treegrid: [InfiniteScroll, Page]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/scroll/virtualscroll-cs3" %}

Limitations for Infinite Scrolling

- Due to the element height limitation in browsers, the maximum number of records loaded by the treegrid is limited due to the browser capability.
- Initial loading rows total height must be greater than the viewport height.
- Cell selection will not be persisted in cache mode.
- Infinite scrolling is not compatible with batch editing, cell editing, detail template and hierarchy features.
- The aggregated information and total group items are displayed based on the current view items. To get these information regardless of the view items, refer to the
- Programmatic selection using the selectRows and selectRow method is not supported in infinite scrolling.

Selection

Selection in Vue Treegrid component

Selection provides an option to highlight a row or cell.

Selection can be done through simple Mouse down or Arrow keys.

To disable selection in the TreeGrid, set the allowSelection to false.

The treegrid supports two types of selection that can be set by using the

selectionSettings.type.They are:

- **Single** - The Single value is set by default. Allows you to select only a single row or cell.
- **Link to the Video** - Allows you to select multiple rows or cells.

To perform the multi-selection, press and hold CTRL key and click the desired rows or cells.

To select range of rows or cells, press and hold the SHIFT key and click the rows or cells.

To get start quickly with selection options, you can check on this video:

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :allowPaging='true'
:selectionSettings='selectionOptions'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
```

```
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      pageSettings: { pageSize: 7 },
      selectionOptions: { type: 'Multiple' }
    };
  },
  provide: {
      treegrid: [ Page ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/selection/default-cs5" %}

Selection mode

TreeGrid supports three types of selection mode which can be set by using

selectionSettings.mode. They are:

- **Row** - The row value is set by default. Allows you to select rows only.
- **Cell** - Allows you to select cells only.
- **Both** - Allows you to select rows and cells at the same time.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :allowPaging='true'
:selectionSettings='selectionOptions' >
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      selectionOptions: { mode: 'Both' }
    };
  },
  provide: {
      treegrid: [ Page ]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/selection/default-cs6" %}

Cell selection in Vue Treegrid component

Cell Selection can be done through simple Mouse down or Arrow keys(up, down, left and right).

TreeGrid supports two types of cell selection mode which can be set by using

selectionSettings.cellSelectionMode. They are:

- **Flow** - The Flow value is set by default. Select range of cells between the start index and end index which includes in between cells of rows.
- **Box** - Select range of cells within the start and end column indexes which includes in between cells of rows within the range.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:selectionSettings='selectionOptions'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { TreeGridPlugin, TreeGridComponent,  Page } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
let initialGridLoad = true;
export default {
  data ()  {
    return {
      data: sampleData,
      selectionOptions: { cellSelectionMode: 'Box', type: 'Multiple', mode:
'Cell' }
    };
  },
  provide: {
      treegrid: [ Page ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/selection/default-cs1" %}

Cell Selection requires the selectionSettings.mode to be Cell or Both and type should be Multiple.

Row selection in Vue Treegrid component

Toggle selection

The Toggle selection allows to perform selection and unselection of the particular row or cell. To enable toggle selection, set enableToggle property of the selectionSettings as true. If you click on the selected row or cell then it will be unselected and vice versa.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1'
        :selectionSettings='selectionOptions'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, TreeGridComponent,  Page } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      selectionOptions: { enableToggle: true}
    };
  },
  provide: {
      treegrid: [ Page ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/selection/default-cs4" %}

If multi selection is enabled, then first click on any selected row (without pressing Ctrl key), it will clear the multi selection and in second click on the same row, it will be unselected.

Check box selection in Vue Treegrid component

Checkbox Selection provides an option to select multiple TreeGrid records with help of checkbox in each row.

To render checkbox in each treegrid row, you need to use checkbox column with type as CheckBox using

column type property.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='2'>
            <e-columns>
                <e-column type='checkbox' width='50'></e-column>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, TreeGridComponent,  Page } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
```

```
    return {
      data: sampleData,
    };
  },
  provide: {
      treegrid: [ Page ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/selection/default-cs2" %}

By default selection is allowed by clicking a treegrid row or checkbox in that row. To allow Selection only through checkbox, you can set

selectionSettings.checkboxOnly property to true.

Selection can be persisted on all the operations

using selectionSettings.persistSelection property.

For persisting selection on the TreeGrid, any one of the column should be defined as a primary key

using columns.isPrimaryKey property.

Checkbox selection mode

In checkbox selection, selection can also be done by clicking on rows. This selection provides two types of Checkbox Selection mode which can be set by using the following API,

selectionSettings.checkboxMode. The modes are;

- **Default**: This is the default value of the checkboxMode. In this mode, user can select multiple rows by clicking rows one by one.
- **ResetOnRowClick**: In ResetOnRowClick mode, when user clicks on a row it will reset previously selected row. Also you can perform multiple-selection in this mode by press and hold CTRL key and click the desired rows. To select range of rows, press and hold the SHIFT key and click the rows.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='2'
        :selectionSettings='selectionOptions'>
            <e-columns>
                <e-column type='checkbox' width='50'></e-column>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, TreeGridComponent,  Page } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      selectionOptions: { checkboxMode: 'ResetOnRowClick'}
    };
  },
  provide: {
      treegrid: [ Page ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/selection/default-cs3" %}

Checkbox Selection feature is intended for row selection only; it is not compatible with cell selection mode.

Aggregates

Aggregates in Vue Treegrid component

Aggregate values are displayed in the TreeGrid footer and in parent row footer for child row aggregate values. It can be configured through `aggregates` property. field and type are the minimum properties required to represent an aggregate column.

To use the aggregate feature, you have to inject the `Aggregate` module.

By default, the aggregate value can be displayed in the treegrid footer, and footer of child rows. To show the aggregate value in one of the cells, use the Link to the Video.

To get start quickly with aggregates in Vue tree grid component, you can check on this video:

Built-in aggregate types

The aggregate type should be specified in the type property to configure an aggregate column.

The built-in aggregates are,

- Sum
- Average
- Min
- Max
- Count
- Truecount
- Falsecount

* Multiple aggregates can be used for an aggregate column by setting the type property

Copyright © 2001 -2024 Syncfusion Inc.

with an array of aggregate types.

* Multiple types for a column is supported only when one of the aggregate templates is used.

Child aggregate

Aggregate value is calculated for child rows, and it is displayed in the parent row footer. Use the childSummary property to render the child rows aggregate value.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='children'
:treeColumnIndex='1' height='260px'>
            <e-columns>
                <e-column field='FreightID' headerText='Freight ID' width=90
textAlign='Right'></e-column>
                <e-column field='FreightName' headerText='Freight Name'
width=180></e-column>
                <e-column field='UnitWeight' headerText='Unit Per Weight'
width=90 type='number'textAlign='Right'></e-column>
                <e-column field='TotalUnits' headerText='Total Units'
type='number' width=80 textAlign='Right'></e-column>
            </e-columns>
              <e-aggregates>
                <e-aggregate :showChildSummary='true'>
                    <e-columns>
                        <e-column type="Max" field="UnitWeight"
:footerTemplate='footerMax'></e-column>
                        <e-column type="Min" field="TotalUnits"
:footerTemplate='footerMin'></e-column>
                    </e-columns>
                </e-aggregate>
          </e-aggregates>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Aggregate } from "@syncfusion/ej2-vue-treegrid";
import { summaryRowData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: summaryRowData,
            footerMin: function () {
        return  { template : Vue.component('minTemplate', {
            template: `<span>Min: {{data.Min}}</span>`,
            data () {return { data: {}};}
            })
          }
      },
      footerMax: function () {
        return  { template : Vue.component('maxTemplate', {
            template: `<span>Max: {{data.Max}}</span>`,
            data () {return { data: {}};}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            })
          }
        }
      };
  },
    provide: {
      treegrid: [Aggregate]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/aggregates/default-cs1" %}

Footer aggregate in Vue Treegrid component

Footer aggregate value is calculated for all the rows, and it is displayed in the footer cells. Use the footerTemplate property to render the aggregate value in footer cells.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='children'
:treeColumnIndex='1' height='260px'>
            <e-columns>
                <e-column field='FreightID' headerText='Freight ID' width=90
textAlign='Right'></e-column>
                <e-column field='FreightName' headerText='Freight Name'
width=180></e-column>
                <e-column field='UnitWeight' headerText='Unit Per Weight'
width=90 type='number' textAlign='Right'></e-column>
                <e-column field='TotalUnits' headerText='Total Units'
type='number' width=80 textAlign='Right'></e-column>
            </e-columns>
              <e-aggregates>
                <e-aggregate :showChildSummary='false'>
                    <e-columns>
                        <e-column type="Max" field="UnitWeight"
:footerTemplate='footerMax'></e-column>
                        <e-column type="Min" field="TotalUnits"
:footerTemplate='footerMin'></e-column>
                    </e-columns>
                </e-aggregate>
          </e-aggregates>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Aggregate } from "@syncfusion/ej2-vue-treegrid";
import { summaryRowData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: summaryRowData,
            footerMin: function () {
```

```
        return  { template : Vue.component('minTemplate', {
            template: `<span>Min: {{data.Min}}</span>`,
            data () {return { data: {}};}
            })
          }
      },
      footerMax: function () {
        return  { template : Vue.component('maxTemplate', {
            template: `<span>Max: {{data.Max}}</span>`,
            data () {return { data: {}};}
            })
          }
      }
    };
  },
    provide: {
      treegrid: [Aggregate]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/aggregates/default-cs2" %}

The aggregate values must be accessed inside the template using their corresponding type name.

How to format aggregate value

You can format the aggregate value result by using the format property.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" childMapping='subtasks'
:treeColumnIndex='1' height='260px'>
            <e-columns>
                <e-column field='category' headerText='Category' width=160
textAlign='Right'></e-column>
                <e-column field='units' headerText='Total Units' width=130
type='number'></e-column>
                <e-column field='unitPrice' headerText='Unit Price($)'
width=110 type='number' format= 'C2' textAlign='Right'></e-column>
                <e-column field='price' headerText='Price($)' type='number'
width=160 format= 'C2' textAlign='Right'></e-column>
            </e-columns>
              <e-aggregates>
                <e-aggregate>
                    <e-columns>
                        <e-column type="Sum" field="price"  format="C2"
:footerTemplate='footerSum'></e-column>
                    </e-columns>
                </e-aggregate>
          </e-aggregates>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { TreeGridPlugin, Aggregate } from "@syncfusion/ej2-vue-treegrid";
import { summaryData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: summaryData,
            footerSum: function () {
        return  { template : Vue.component('minTemplate', {
            template: `<span>Sum: {{data.Sum}}</span>`,
            data () {return { data: {}};}
            })
          }
      }
    };
  },
    provide: {
      treegrid: [Aggregate]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/aggregates/default-cs3" %}

Print in Vue Treegrid component

To print the TreeGrid, use the print method from treegrid instance. The print option can be displayed on the toolbar by adding the print toolbar item.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' height='265'
childMapping='subtasks' :treeColumnIndex='1' :toolbar='toolbarOptions'>
            <e-columns>
              <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
              <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
              <e-column field='startDate' headerText='Start Date' width='90'
format="yMd" textAlign='Right'></e-column>
              <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Toolbar } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
```

```
      toolbarOptions: ['Print']
    };
  },
  provide: {
    treegrid: [ Toolbar ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/print/default-cs1" %}

Page setup

Some of the print options cannot be configured through JavaScript code. So, you have to customize the layout, paper size, and margin options using the browser page setup dialog. Please refer to the following links to know more about the browser page setup:

- Chrome
- Firefox
- Safari
- IE

Print using an external button

To print the treegrid from an external button, invoke the print method.

APP.VUE

```
<template>
<div id="app">
        <ejs-button id='print' @click.native='print'>Print</ejs-button>
        <ejs-treegrid ref='treegrid' :dataSource='data' height='265'
childMapping='subtasks' :treeColumnIndex='1'>
            <e-columns>
              <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
              <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
              <e-column field='startDate' headerText='Start Date' width='90'
format="yMd" textAlign='Right'></e-column>
              <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
Vue.use(ButtonPlugin);
export default {
  data ()  {
    return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      data: sampleData,
    };
  },
  methods: {
    print: function() {
        this.$refs.treegrid.print();
    }
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/print/default-cs2" %}

Print visible Page

By default, the grid prints all the pages. To print the current page alone, set the printMode to CurrentPage.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' printMode='CurrentPage'
childMapping='subtasks' :treeColumnIndex='1' :toolbar='toolbarOptions'
:allowPaging='true' :pageSettings='pageSettings'>
            <e-columns>
              <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
              <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
              <e-column field='startDate' headerText='Start Date' width='90'
format="yMd" textAlign='Right'></e-column>
              <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['Print'],
      pageSettings: { pageSize: 8 },
    };
  },
  provide: {
    treegrid: [ Page, Toolbar ]
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/treegrid/print/default-cs3" %}

Print large number of columns

By default, the browser uses A4 as page size option to print pages and to adapt the size of the page the browser print preview will auto-hide the overflowed contents. Hence treegrid with large number of columns will cut off to adapt the print page.

To show large number of columns when printing, adjust the scale option from print option panel based on your content size.

Show or Hide columns while Printing

You can show a hidden column or hide a visible column while printing the treegrid using toolbarClick and printComplete events.

In the toolbarClick event, based on args.item.text as Print. We can show or hide columns by setting column.visible property to true or false respectively.

In the printComplete event, We have reversed the state back to the previous state.

In the below example, we have Duration as a hidden column in the treegrid. While printing, we have changed Duration to visible column and StartDate as hidden column.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
printMode='CurrentPage' childMapping='subtasks' :treeColumnIndex='1'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'
:printComplete='printComplete'>
            <e-columns>
              <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
```

```
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date' width='90'
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
:visible='false' textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['Print'],
    };
  },
  methods: {
      toolbarClick: function() {
          for (var i = 0; i <
this.$refs.treegrid.ej2Instances.grid.getColumns().length; i++) {
            if (this.$refs.treegrid.ej2Instances.grid.getColumns()[i].field
== "duration") {
              this.$refs.treegrid.ej2Instances.grid.getColumns()[i].visible
= true;
            }
            else if
(this.$refs.treegrid.ej2Instances.grid.getColumns()[i].field == "startDate")
{
              this.$refs.treegrid.ej2Instances.grid.getColumns()[i].visible
= false;
            }
        }
      },
      printComplete: function() {
          for (var i = 0; i < this.$refs.grid.getColumns().length; i++) {
            if (this.$refs.treegrid.grid.getColumns()[i].field ==
"duration") {
              this.$refs.treegrid.grid.getColumns()[i].visible = false;
            }
            else if (this.$refs.treegrid.grid.getColumns()[i].field ==
"startDate") {
              this.$refs.treegrid.grid.getColumns()[i].visible = true;
            }
        }
      }
  },
  provide: {
    treegrid: [ Page, Toolbar ]
  }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/print/default-cs4" %}

Limitations of Printing Large Data

When treegrid contains large number of data, printing all the data at once is not a best option for the browser performance. Because to render all the DOM elements in one page will produce performance issues in the browser. It leads to browser slow down or browser hang.

If printing of all the data is still needed, we suggest to Export the treegrid to Excel or CSV or Pdf and then print it from another non-web based application.

Adaptive in Vue Treegrid component

The TreeGrid user interface (UI) was redesigned to provide an optimal viewing experience and improve usability on small screens. This interface will render the filter, sort, and edit dialogs adaptively.

Render adaptive dialogs

When you enable the enableAdaptiveUI property, the tree grid will render the filter, sort, and edit dialogs in full screen for a better user experience. The following demo demonstrates this behaviour.

APP.VUE

```
<template>
    <div id="app">
    <div class="e-bigger e-adaptive-demo">
    <div class="e-mobile-layout">
        <div class="e-mobile-content">
            <ejs-treegrid ref='treegrid' id='adaptivebrowser'
:dataSource='data' childMapping='subtasks' :treeColumnIndex='1'
height='100%' :enableAdaptiveUI='true' :allowPaging='true'
:allowSorting='true' :allowFiltering='true' :editSettings='editSettings'
:toolbar='toolbar' :filterSettings='filterSettings' :load='load'>
                <e-columns>
                    <e-column field='taskID' headerText='Task ID'
:isPrimaryKey='true' width='135' textAlign='Right'></e-column>
                    <e-column field='taskName' headerText='Task Name'
width='280'></e-column>
                    <e-column field='duration' headerText='Duration'
width='140' textAlign='Right'></e-column>
                    <e-column field='progress' headerText='Progress'
width='145' textAlign='Right'></e-column>
                </e-columns>
            </ejs-treegrid>
        </div>
    </div>
    </div>
    </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Edit, Sort, Filter, Toolbar } from
"@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  data() {
    return {
      data: sampleData,
      editSettings: { allowAdding: true, allowEditing: true, allowDeleting:
true, mode: 'Dialog' },
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel', 'Search'],
      filterSettings: { type: 'Excel' }
    };
  },
  methods: {
    load: function() {
        (this.$refs.treegrid as
any).$el.ej2_instances[0].grid.adaptiveDlgTarget =
document.getElementsByClassName('e-mobile-content')[0];
    }
  },
  provide: {
      treegrid: [Page, Edit, Sort, Filter, Toolbar]
  }
}
</script>
<style>
 @import "../node_modules/@syncfusion/ej2-vue-treegrid/styles/material.css";
 /* The device with borders */
    .e-mobile-layout {
        position: relative;
        width: 360px;
        height: 640px;
        margin: auto;
        border: 16px #f4f4f4 solid;
        border-top-width: 60px;
        border-bottom-width: 60px;
        border-radius: 36px;
        box-shadow: 0 0px 2px rgb(144 144 144), 0 0px 10px rgb(0 0 0 / 16%);
    }
    /* The horizontal line on the top of the device */
    .e-mobile-layout:before {
        content: '';
        display: block;
        width: 60px;
        height: 5px;
        position: absolute;
        top: -30px;
        left: 50%;
        transform: translate(-50%, -50%);
        background: #ebebeb;
        border-radius: 10px;
    }
    /* The circle on the bottom of the device */
    .e-mobile-layout:after {
        content: '';
        display: block;
        width: 35px;
        height: 35px;
        position: absolute;
        left: 50%;
        bottom: -65px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        transform: translate(-50%, -50%);
        background: #e8e8e8;
        border-radius: 50%;
    }
    /* The screen (or content) of the device */
    .e-mobile-layout .e-mobile-content {
        overflow-x: hidden;
        width: 328px;
        height: 521px;
        background: white;
        border: 0px solid #dddddd;
    }
    .highcontrast .e-mobile-layout {
        border: 16px #000000 solid;
        border-top-width: 60px;
        border-bottom-width: 60px;
        box-shadow: -1px 2px white, -2px -2px white, 2px -2px white, 2px 1px
white;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/adaptive-cs1" %}

Loading animation in Vue Treegrid component

The Tree Grid displays a loading indicator while the data is being fetched and bound to the tree grid during initial rendering, refreshing, and after performing any tree grid actions like sorting, filtering, and more.

The tree grid supports two indicator types, which can be enabled by setting the `loadingIndicator.indicatorType` property to Spinner or Shimmer. The default value of the indicator type is Spinner.

In the following sample, the Shimmer indicator is displayed while the tree grid is loading and refreshing when using the remote data.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" idMapping='TaskID'
parentIdMapping='ParentItem' hasChildMapping='isParent' :treeColumnIndex='1'
:allowPaging='true' :allowSorting='true' :pageSettings='pageSettings'
:loadingIndicator='loadingIndicator'>
            <e-columns>
                <e-column field='TaskID' headerText='Task ID' width=120
textAlign='Right'></e-column>
                <e-column field='TaskName' headerText='Task Name' width=240
textAlign='Left'></e-column>
                <e-column field='StartDate' headerText='Start Date'
width=140 format="yMd" textAlign='Right' type='date'></e-column>
                <e-column field='Duration' headerText='Duration' width=130
textAlign='Right'></e-column>
                <e-column field='Progress' headerText='Progress'
width=130></e-column>
                </e-columns>
        </ejs-treegrid>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Sort } from "@syncfusion/ej2-vue-treegrid";
import { DataManager, WebApiAdaptor } from "@syncfusion/ej2-data";
Vue.use(TreeGridPlugin);
export default {
   data() {
    let SERVICE_URI ="https://ej2services.syncfusion.com/production/web-services/api/SelfReferenceData";
    return {
      data: new DataManager({
        url: SERVICE_URI,
        adaptor: new WebApiAdaptor(),
        crossDomain: true
      }),
      pageSettings: { pageCount: 3 },
      loadingIndicator: { indicatorType: 'Shimmer' },
    };
  }
  provide: {
      treegrid: [ Page, Sort ]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/indicator/default-cs1" %}

State Persistence

State persistence in Vue Treegrid component

State persistence refers to the TreeGrid's state maintained in the browser's localStorage even if the browser is refreshed or if you move to the next page within the browser.

State persistence stores treegrid's model object in the local storage when the enablePersistence is defined as true.

Get or set local storage value in Vue Treegrid component

If the enablePersistence property is set to true, the treegrid property value is saved in the window.localStorage for reference. You can get/set the localStorage value by using the getItem/setItem method in the window.localStorage.

`ts

//get the TreeGrid model.

let value: string = window.localStorage.getItem('treegridTreeGrid'); //"treegridTreeGrid" is component name + component id.

let model: Object = JSON.parse(model);

`

`ts

//set the TreeGrid model.

Copyright © 2001 -2024 Syncfusion Inc.

window.localStorage.setItem('treegridTreeGrid', JSON.stringify(model)); //"treegridTreeGrid" is component name + component id.

`

Tool Bar

Tool bar in Vue Treegrid component

The TreeGrid provides ToolBar support to handle treegrid actions. The toolbar

property accepts either the collection of built-in toolbar items and ItemModelobjects for custom toolbar items or HTML element ID for toolbar template.

To use ToolBar, inject Toolbar module in the treegrid.

Enable/disable toolbar items

You can enable/disable toolbar items by using the enableItems method.

APP.VUE

```
<template>
<div id="app">
      <ejs-button id='enable' cssClass='e-flat'
@click.native='enable'>Enable</ejs-button>
      <ejs-button id='disable' cssClass='e-flat'
@click.native='disable'>Disable</ejs-button>
      <ejs-treegrid ref='treegrid' :dataSource='data' height='200'
childMapping='subtasks' :treeColumnIndex='1' :toolbar='toolbar'
:toolbarClick='toolbarClick' :allowFiltering='true'>
            <e-columns>
              <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
              <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
              <e-column field='startDate' headerText='Start Date' width='90'
format="yMd" textAlign='Right'></e-column>
              <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
      </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Toolbar, Filter } from "@syncfusion/ej2-vue-
treegrid";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { EmitType } from '@syncfusion/ej2-base';
import { sampleData } from "./datasource.js";
import { ClickEventArgs } from '@syncfusion/ej2-vue-navigations';
Vue.use(TreeGridPlugin);
Vue.use(ButtonPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbar: ['QuickFilter', 'ClearFilter'],
    };
```

```
  },
  methods: {
    toolbarClick: function (args: ClickEventArgs) {
      if (args.item.text === 'QuickFilter') {
        this.$refs.treegrid.filterByColumn('taskName', 'startswith',
'Testing');
      }
      if (args.item.text === 'ClearFilter') {
        this.$refs.treegrid.clearFiltering();
      }
    },
    enable: function() {

this.$refs.treegrid.ej2Instances.toolbarModule.enableItems(['_gridcontrol_Qu
ickFilter', '_gridcontrol_ClearFilter'], true);//Enable toolbar items.
    },
    disable: function() {

this.$refs.treegrid.ej2Instances.toolbarModule.enableItems(['_gridcontrol_Qu
ickFilter', '_gridcontrol_ClearFilter'], false);//Disable toolbar items.
    }
  },
  provide: {
      treegrid: [ Toolbar, Filter ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/toolbar/default-cs4" %}

Tool bar items in Vue Treegrid component

Built-in toolbar items

The following table shows built-in toolbar items and its actions.

Built-in Toolbar Items	Actions
ExpandAll	Expands all the rows.
CollapseAll	Collapses all the rows.
Add	Adds a new record.
Edit	Edits the selected record.
Update	Updates the edited record.
Delete	Deletes the selected record.
Cancel	Cancels the edit state.
Search	Searches the records by the given key.
Print	Prints the treegrid.
ExcelExport	Exports the treegrid to Excel.
PdfExport	Exports the treegrid to PDF.

Copyright © 2001 -2024 Syncfusion Inc.

| WordExport | Exports the treegrid to Word.|

| Indent | Indents the record to one level of hierarchy.|

| Outdent | Outdents the record to one level of hierarchy.|

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' height='265'
childMapping='subtasks' :treeColumnIndex='1' :toolbar='toolbar'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Toolbar, Filter } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbar: ['Print', 'Search']
    };
  },
  provide: {
      treegrid: [ Toolbar, Filter ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/toolbar/default-cs1" %}

* The toolbar has options to define both built-in and custom toolbar items.

Custom toolbar items

Custom toolbar items can be added by defining the toolbar as a collection of

ItemModels. Actions for this customized toolbar items are defined in the toolbarClick event.

By default, Custom toolbar items are in position Left. You can change the position by using the align property. In the below sample, we have applied position Right for the Quick Filter toolbar item.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :toolbar='toolbar'
:allowFiltering='true' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Toolbar, Filter } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
import { ClickEventArgs } from '@syncfusion/ej2-vue-navigations';
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbar: [{text: 'Quick Filter', tooltipText: 'Quick Filter', id:
'toolbarfilter', align:'Right'}]
    };
  },
  methods: {
    toolbarClick: function (args: ClickEventArgs) {
      if (args.item.id === 'toolbarfilter') {
        this.$refs.treegrid.filterByColumn('taskName', 'startswith',
'Testing');
      }
    }
  },
  provide: {
      treegrid: [ Toolbar, Filter ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/toolbar/default-cs2" %}

* The toolbar has options to define both built-in and custom toolbar items.

* If a toolbar item does not match the built-in items, it will be treated as a custom toolbar item.

Built-in and custom items in toolbar

TreeGrid have an option to use both built-in and custom toolbar items at same time.

Copyright © 2001 -2024 Syncfusion Inc.

In the below example, ExpandAll, CollapseAll are built-in toolbar items and Click is custom toolbar item.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :toolbar='toolbar'
:toolbarClick='toolbarClick' :editSettings='editSettings'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='180'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Toolbar, Edit } from "@syncfusion/ej2-vue-
treegrid";
import { EmitType } from '@syncfusion/ej2-base';
import { sampleData } from "./datasource.js";
import { ClickEventArgs } from '@syncfusion/ej2-vue-navigations';
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbar: ['ExpandAll', 'CollapseAll', { text: 'Click', tooltipText:
'Click', prefixIcon: 'e-time', id: 'Click' }],
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
    };
  },
  methods: {
    toolbarClick: function (args: ClickEventArgs) {
      if (args.item.text === 'Click') {
        alert("Custom toolbar click...");
      }
    }
  },
  provide: {
      treegrid: [ Toolbar, Edit ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/toolbar/default-cs3" %}

Copyright © 2001 -2024 Syncfusion Inc.

Pdf Export

Pdf export in Vue Treegrid component

PDF export allows exporting TreeGrid data to PDF document. You need to use the pdfExport method for exporting. To enable PDF export in the treegrid, set the allowPdfExport as true.

To use PDF export, inject the Link to the Video module in treegrid.

To get start quickly with Pdf export feature, you can check on this video:

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowPdfExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['PdfExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'PDF Export') {
            this.$refs.treegrid.pdfExport();
        }
    }
  },
  provide: {
      treegrid: [ Page, Toolbar, PdfExport ]
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/treegrid/pdf/default-cs10" %}

Custom data source

PDF export provides an option to define datasource dynamically before exporting. To export data dynamically, define the dataSource in exportProperties.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowPdfExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['PdfExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'PDF Export') {
          let pdfExportProperties = {
            dataSource: sampleData,
          }
          this.$refs.treegrid.pdfExport(pdfExportProperties);
        }
      }
  },
  provide: {
      treegrid: [ Page, Toolbar, PdfExport ]
  }
}
```

```
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/pdf/default-cs11" %}

Pdf export options in Vue Treegrid component

Export hidden columns

PDF export provides an option to export hidden columns of TreeGrid by defining the includeHiddenColumn as true.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowPdfExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration'
:visible='false' width='80' textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['PdfExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'PDF Export') {
          let pdfExportProperties = {
            includeHiddenColumn: true
          }
          this.$refs.treegrid.pdfExport(pdfExportProperties);
        }
    }
  },
  provide: {
      treegrid: [ Page, Toolbar, PdfExport ]
```

```
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/pdf/default-cs4" %}

Show or hide columns on exported PDF

You can show a hidden column or hide a visible column while exporting the treegrid using toolbarClick and pdfExportComplete events.

In the toolbarClick event, based on args.item.text as PDF Export. We can show or hide columns by setting column.visible property to true or false respectively.

In the pdfExportComplete event, We have reversed the state back to the previous state.

In the below example, we have Duration as a hidden column in the treegrid. While exporting, we have changed Duration to visible column and StartDate as hidden column.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowPdfExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'
:pdfExportComplete='pdfExportComplete'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
:visible='false' textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['PdfExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
```

```
        if (args['item'].text === 'PDF Export') {
          this.$refs.treegrid.ej2Instances.grid.getColumns()[2].visible =
false;
          this.$refs.treegrid.ej2Instances.grid.getColumns()[3].visible =
true;
          this.$refs.treegrid.pdfExport();
        }
      }
      pdfExportComplete(args) {
        this.$refs.treegrid.ej2Instances.grid.getColumns()[2].visible =
true;
        this.$refs.treegrid.ej2Instances.grid.getColumns()[3].visible =
false;
      }
  },
  provide: {
      treegrid: [ Page, Toolbar, PdfExport ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/pdf/default-cs5" %}

How to change page orientation

Page orientation can be changed Landscape(Default Portrait) for the exported document using the exportProperties.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowPdfExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
```

```
    return {
      data: sampleData,
      toolbarOptions: ['PdfExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'PDF Export') {
          let pdfExportProperties = {
            pageOrientation: 'Landscape',
          }
          this.$refs.treegrid.pdfExport(pdfExportProperties);
        }
    }
  },
  provide: {
      treegrid: [ Page, Toolbar, PdfExport ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/pdf/default-cs6" %}

How to change page size

Page size can be customized for the exported document using the exportProperties.

Supported page sizes are:

- Letter
- Note
- Legal
- A0
- A1
- A2
- A3
- A5
- A6
- A7
- A8
- A9
- B0
- B1
- B2
- B3
- B4
- B5
- Archa
- Archb
- Archc
- Archd

Copyright © 2001 -2024 Syncfusion Inc.

- Arche
- Flsa
- HalfLetter
- Letter11x17
- Ledger

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowPdfExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['PdfExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'PDF Export') {
          let pdfExportProperties = {
            pageSize: 'Letter'
          }
          this.$refs.treegrid.pdfExport(pdfExportProperties);
        }
    }
  },
  provide: {
      treegrid: [ Page, Toolbar, PdfExport ]
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/treegrid/pdf/default-cs7" %}

To customize PDF export

PDF export provides an option to customize mapping of treegrid to exported PDF document.

File name for exported document

You can assign the file name for the exported document by defining `fileName` property in `PdfExportProperties`.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowPdfExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['PdfExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'PDF Export') {
          let pdfExportProperties = {
            fileName:"new.xlsx"
          }
          this.$refs.treegrid.pdfExport(pdfExportProperties);
        }
    }
  },
  provide: {
```

```
      treegrid: [ Page, Toolbar, PdfExport ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/pdf/default-cs8" %}

Default fonts for PDF exporting

By default, treegrid uses Helvetica font in the exported document. You can change the default font by using pdfExportProperties.theme property. The available default fonts are,

- Helvetica
- TimesRoman
- Courier
- Symbol
- ZapfDingbats

The code example for changing default font,

`ts

import { PdfStandardFont, PdfFontFamily, PdfFontStyle } from '@syncfusion/ej2-pdf-export';

...

let pdfExportProperties: PdfExportProperties = {

theme: {

header: {font: new PdfStandardFont(PdfFontFamily.TimesRoman, 11, PdfFontStyle.Bold),

record: { font: new PdfStandardFont(PdfFontFamily.TimesRoman, 10) }

}

};

`

Add custom font for PDF exporting

You can change the default font of TreeGrid header, content and caption cells in the exported document by using pdfExportProperties.theme property.

In the following example, we have used Advent Pro font to export the treegrid with Hungarian fonts.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowPdfExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData, adventProFont } from "./datasource.js";
import { PdfTrueTypeFont } from '@syncfusion/ej2-pdf-export';
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['PdfExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'PDF Export') {
          let pdfExportProperties = {
            theme: {
                header: {font:  new PdfTrueTypeFont(adventProFont, 12) },
                record: { font: new PdfTrueTypeFont(adventProFont, 9) }
            }
          }
          this.$refs.treegrid.pdfExport(pdfExportProperties);
        }
    }
  },
  provide: {
      treegrid: [ Page, Toolbar, PdfExport ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/pdf/default-cs9" %}

PdfTrueTypeFont accepts base 64 format of the Custom Font.

Pdf cell style customization in Vue Treegrid component

Conditional cell formatting

TreeGrid cells in the exported PDF can be customized or formatted using pdfQueryCellInfo event. In this event, we can format the treegrid cells of exported PDF document based on the column cell value.

In the below sample, we have set the background color for Duration column in the exported document by args.cell and backgroundColor property.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowPdfExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'
:queryCellInfo='queryCellInfo'
        :pdfQueryCellInfo='pdfQueryCellInfo'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['PdfExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'PDF Export') {
          this.$refs.treegrid.pdfExport();
        }
      }
      pdfQueryCellInfo(args) {
        if(args.column.field == 'duration'){
          if(+args.value === 0 || args.value === "") {
            args.style = {backgroundColor: '#336c12'};
          }
          else if(args.value < 3) {
            args.style = {backgroundColor: '#7b2b1d'};
          }
        }
      }
      queryCellInfo(args) {
        if (args.data['duration'] == 0 && args.column.field === 'duration' )
{
            args.cell.style.background= '#336c12';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        } else if (args.data['duration'] < 3 && args.column.field ===
'duration') {
            args.cell.style.background= '#7b2b1d';
        }
      }
  },
  provide: {
      treegrid: [ Page, Toolbar, PdfExport ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/pdf/default-cs2" %}

Theme

PDF export provides an option to include theme for exported PDF document.

To apply theme in exported PDF, define the theme in exportProperties.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowPdfExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['PdfExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'PDF Export') {
```

```
            let pdfExportProperties = {
              theme: {
                  header: {
                      fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true, borders: { color: '#64FA50', lineStyle: 'Thin' }
                  },
                  record: {
                      fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true
                  }
              }
            }
            this.$refs.treegrid.pdfExport(pdfExportProperties);
          }
        }
    },
    provide: {
        treegrid: [ Page, Toolbar, PdfExport ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/pdf/default-cs3" %}

By default, material theme is applied to exported PDF document.

Adding header and footer in Vue Treegrid component

You can customize text, page number, line, page size and changing orientation in header and footer.

How to write a text in header or footer

You can add text either in Header or Footer of exported PDF document.

`ts

let exportProperties: PdfExportProperties = {

header: {

fromTop: 0,

height: 130,

contents: [

{

type: 'Text',

value: "Task Details",

position: { x: 0, y: 50 },

style: { textBrushColor: '#000000', fontSize: 13 }

},

]

}

`

How to draw a line in header or footer

You can add line either in Header or Footer of the exported PDF document.

Supported line styles:

- dash
- dot
- dashdot
- dashdotdot
- solid

`ts

let exportProperties: PdfExportProperties = {

header: {

fromTop: 0,

height: 130,

contents: [

{

type: 'Line',

style: { penColor: '#000080', penSize: 2, dashStyle: 'Solid' },

points: { x1: 0, y1: 4, x2: 685, y2: 4 }

}

]

}

}

`

Add page number in header or footer

you can add page number either in Header or Footer of exported PDF document.

Supported page number types:

- LowerLatin - a, b, c,
- UpperLatin - A, B, C,
- LowerRoman - i, ii, iii,
- UpperRoman - I, II, III,
- Number - 1,2,3.

`ts

let exportProperties: PdfExportProperties = {

header: {

Copyright © 2001 -2024 Syncfusion Inc.

```
fromTop: 0,
height: 130,
contents: [
{
type: 'PageNumber',
pageNumberType: 'Arabic',
format: 'Page {$current} of {$total}', //optional
position: { x: 0, y: 25 },
style: { textBrushColor: '#ffff80', fontSize: 15, hAlign: 'Center' }
}
]
}
}
```

Insert an image in header or footer

Image (Base64 string) can be added in the exported document in header/footer using the exportProperties.

```ts
let exportProperties: PdfExportProperties = {
header: {
fromTop: 0,
height: 130,
contents: [
{
type: 'Image',
src: image,
position: { x: 40, y: 10 },
size: { height: 100, width: 250 },
}
]
}
}
```

The below code illustrates the pdf export customization.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowPdfExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
import { image } from './image.js';
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['PdfExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'PDF Export') {
          let pdfExportProperties = {
            header: {
                fromTop: 0,
                height: 130,
                contents: [
                    {
                        type: 'Image',
                        src: image,
                        position: { x: 40, y: 10 },
                        size: { height: 100, width: 250 },
                    }
                ]
            },
            footer: {
                fromBottom: 160,
                height: 150,
                contents: [
                    {
```

```
                        type: 'PageNumber',
                        pageNumberType: 'Arabic',
                        format: 'Page {$current} of {$total}',
                        position: { x: 0, y: 25 },
                        style: { textBrushColor: '#ffff80', fontSize: 15 }
                    }
                ]
            }
        }
        this.$refs.treegrid.pdfExport(pdfExportProperties);
    }
  }
 },
 provide: {
     treegrid: [ Page, Toolbar, PdfExport ]
 }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/pdf/default-cs1" %}

Exporting tree grid in server in Vue Tree Grid component

The Tree Grid have an option to export the data to PDF in server side using tree grid server export library.

Server dependencies

The Server side export functionality is shipped in the Syncfusion.EJ2.TreeGridExport package, which is available in Essential Studio and nuget.org. The following list of dependencies is required for tree grid server side PDF exporting action.

- Syncfusion.EJ2
- Syncfusion.EJ2.TreeGridExport

Server configuration

The following code snippet shows server configuration using ASP.NET Core Controller Action.

To Export the tree grid in server side, You need to call the serverPdfExport method for passing the tree grid properties to server exporting action.

`ts

public IActionResult ServerSideExporting()

{

var order = TreeData.GetDefaultData();

ViewBag.dataSource = order;

return View();

}

public IActionResult PdfExport(string treeGridModel)

{

```
if (treeGridModel == null)
{
return View();
}
TreeGridPdfExport exp = new TreeGridpdfExport();
Syncfusion.EJ2.TreeGrid.TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
return exp.ExportToPdf<TreeData>(gridProperty, TreeData.GetDefaultData());
}
private Syncfusion.EJ2.TreeGrid.TreeGrid ConvertTreeGridObject(string gridProperty)
{
Syncfusion.EJ2.TreeGrid.TreeGrid TreeGridModel =
(Syncfusion.EJ2.TreeGrid.TreeGrid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
typeof(Syncfusion.EJ2.TreeGrid.TreeGrid));
TreeGridColumnModel cols =
(TreeGridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
typeof(TreeGridColumnModel));
TreeGridModel.Columns = cols.columns;
return TreeGridModel;
}
public class TreeGridColumnModel
{
public List<TreeGridColumn> columns { get; set; }
}
`
```

```
`ts
<template>
<div id="app">
<ejs-treegrid ref='treegrid' :dataSource='data' height='220' childMapping='subtasks'
:treeColumnIndex='1' :allowPdfExport='true' :toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
<e-columns>
<e-column field='taskID' headerText='Task ID' width='90' textAlign='Right'></e-column>
<e-column field='taskName' headerText='Task Name' width='160'></e-column>
<e-column field='startDate' headerText='Start Date' width='90' format="yMd" textAlign='Right'></e-
column>
<e-column field='duration' headerText='Duration' width='80' textAlign='Right'></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</e-columns>
</ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
data ()  {
return {
data: sampleData,
toolbarOptions: ['PdfExport'],
};
},
methods: {
toolbarClick(args) {
if (args['item'].text === 'Pdf Export') {
this.$refs.treegrid.serverExcelExport('Home/PdfExport');
}
}
},
provide: {
treegrid: [ Page, Toolbar, PdfExport ]
}
}
</script>
`
```

Rotate a header text to a certain degree in the exported tree grid on the server side

The Tree Grid has support to customize the column header styles such as changing text orientation, the font color, and so on in the exported PDF file. To achieve this requirement, define the BeginCellLayout event of the PdfExportProperties with an event handler to perform the required action.

Copyright © 2001 -2024 Syncfusion Inc.

The PdfHeaderCellRendering will be triggered when creating a column header for the pdf document to be exported. Collect the column header details in this event and handle the custom in the BeginCellLayout event handler.

In the following demo, the DrawString method from the Graphics is used to rotate the header text of the column header inside the BeginCellLayout event handler.

A PDF exporting is not supported to rotate the column header on the client side.

`ts

```
public IActionResult PdfExport(string treeGridModel)
{
if (treeGridModel == null)
{
return View();
}
TreeGridPdfExport exp = new TreeGridPdfExport();
TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
gridProperty.PdfHeaderCellRendering = PdfHeaderQueryCellInfo;
PdfGrid grid = new PdfGrid();
Syncfusion.EJ2.TreeGridExport.PdfExportProperties pdfExportProperties = new
Syncfusion.EJ2.TreeGridExport.PdfExportProperties();
pdfExportProperties.IsRepeatHeader = true;
exp.BeginCellLayout = new PdfGridBeginCellLayoutEventHandler(BeginCellEvent);
System.Collections.IEnumerable data = Syncfusion.EJ2.Base.Dat.DataTableToJson(ViewBag.dataSource);
var result = exp.PdfExport<dynamic>(gridProperty, data, pdfExportProperties);
return View();
}
public void BeginCellEvent(object sender, PdfGridBeginCellLayoutEventArgs args)
{
PdfGrid grid = (PdfGrid)sender;
var brush = new PdfSolidBrush(new PdfColor(Color.DimGray));
args.Graphics.Save();
args.Graphics.TranslateTransform(args.Bounds.X + 50, args.Bounds.Height + 40); // give the value for
bounds x and Y by the user
args.Graphics.RotateTransform(-60);   // give the rotate degree value by the user
// Draw the text at particular bounds.
```

Copyright © 2001 -2024 Syncfusion Inc.

```
args.Graphics.DrawString(headerValues[args.CellIndex], new PdfStandardFont(PdfFontFamily.Helvetica, 10), brush, new PointF(0, 0));

if (args.IsHeaderRow)

{

grid.Headers[0].Cells[args.CellIndex].Value = string.Empty;

}

args.Graphics.Restore();

}

private void PdfHeaderQueryCellInfo(object pdf)

{

Syncfusion.EJ2.TreeGridExport.PdfHeaderCellEventArgs name = (Syncfusion.EJ2.TreeGridExport.PdfHeaderCellEventArgs)pdf;

PdfGrid grid = new PdfGrid();

headerValues.Add(name.Column.HeaderText);

var longestString = headerValues.Where(s => s.Length == headerValues.Max(m => m.Length)).First();

PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 6);

SizeF size = font.MeasureString(longestString);

name.Headers[0].Height = size.Width * 2;

}
`
```

Excel Export

Excel export in Vue Treegrid component

The excel export allows exporting TreeGrid data to Excel document. You need to use the excelExport method for exporting. To enable Excel export in the treegrid, set the allowExcelExport as true.

To use excel export, You need to inject the Link to the Video module in treegrid.

To get start quickly with excel export feature, you can check on this video:

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220' childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true' :pageSettings='pageSettings' :allowExcelExport='true' :toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90' textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name' width='160'></e-column>
```

```
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, ExcelExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['ExcelExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'Excel Export') {
            this.$refs.treegrid.excelExport();
        }
    }
  },
  provide: {
      treegrid: [ Page, Toolbar, ExcelExport ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/excel/default-cs8" %}

Custom data source

The excel export provides an option to define datasource dynamically before exporting. To export data dynamically, define the dataSource in exportProperties.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowExcelExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
```

```
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, ExcelExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['ExcelExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'Excel Export') {
          let excelExportProperties = {
            dataSource: sampleData
          }
          this.$refs.treegrid.excelExport(excelExportProperties);
        }
    }
  },
  provide: {
      treegrid: [ Page, Toolbar, ExcelExport ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/excel/default-cs9" %}

Excel export options in Vue Treegrid component

To customize excel export

The excel export provides an option to customize mapping of the treegrid to excel document.

Export hidden columns

The excel export provides an option to export hidden columns of treegrid by defining `includeHiddenColumn` as `true`.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowExcelExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
```

```
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
:visible='false' textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, ExcelExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['ExcelExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'Excel Export') {
          let excelExportProperties = {
            includeHiddenColumn: true
          }
          this.$refs.treegrid.excelExport(excelExportProperties);
        }
    }
  },
  provide: {
      treegrid: [ Page, Toolbar, ExcelExport ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/excel/default-cs4" %}

Show or hide columns on exported excel

You can show a hidden column or hide a visible column while printing the treegrid using toolbarClick and excelExportComplete events.

In the toolbarClick event, based on args.item.text as Excel Export. We can show or hide columns by setting column.visible property to true or false respectively.

In the excelExportComplete event, We have reversed the state back to the previous state.

In the below example, we have Duration as a hidden column in the treegrid. While exporting, we have changed Duration to visible column and StartDate as hidden column.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowExcelExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'
:excelExportComplete='excelExportComplete'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
:visible='false' textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, ExcelExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['ExcelExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'Excel Export') {
          this.$refs.treegrid.ej2Instances.grid.getColumns()[2].visible =
false;
          this.$refs.treegrid.ej2Instances.grid.getColumns()[3].visible =
true;
          this.$refs.treegrid.excelExport();
        }
      }
      excelExportComplete() {
        this.$refs.treegrid.ej2Instances.grid.getColumns()[2].visible =
true;
        this.$refs.treegrid.ej2Instances.grid.getColumns()[3].visible =
false;
      }
  },
  provide: {
      treegrid: [ Page, Toolbar, ExcelExport ]
  }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/excel/default-cs5" %}

File name for exported document

You can assign the file name for the exported document by defining fileName property in ExcelExportProperties.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowExcelExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, ExcelExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['ExcelExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'Excel Export') {
          let excelExportProperties = {
            fileName:"new.xlsx"
          }
          this.$refs.treegrid.excelExport(excelExportProperties);
        }
    }
  },
  provide: {
      treegrid: [ Page, Toolbar, ExcelExport ]
  }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/excel/default-cs6" %}

Persist collapsed state

You can persist the collapsed state in the exported document by defining isCollapsedStatePersist property as true in TreeGridExcelExportProperties parameter of excelExport method.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowExcelExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, ExcelExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['ExcelExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'Excel Export') {
          let excelExportProperties = {
              isCollapsedStatePersist: true
          }
          this.$refs.treegrid.excelExport(excelExportProperties);
        }
    }
  },
  provide: {
      treegrid: [ Page, Toolbar, ExcelExport ]
```

```
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/excel/default-cs7" %}

Excel cell style customization in Vue Treegrid component

Conditional cell formatting

TreeGrid cells in the exported Excel can be customized or formatted using excelQueryCellInfo event. In this event, we can format the treegrid cells of exported PDF document based on the column cell value.

In the below sample, we have set the background color for Duration column in the exported excel by args.cell and backgroundColor property.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowExcelExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'
:queryCellInfo='queryCellInfo'
        :excelQueryCellInfo='excelQueryCellInfo'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, ExcelExport } from "@syncfusion/ej2-
vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['ExcelExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'Excel Export') {
          this.$refs.treegrid.excelExport();
        }
```

```
      }
      excelQueryCellInfo(args) {
        if(args.column.field == 'duration'){
          if(args.value === 0 || args.value === "") {
            args.style = {backColor: '#336c12'};
          }
          else if(args.value < 3) {
            args.style = {backColor: '#7b2b1d'};
          }
        }
      }
      queryCellInfo(args) {
        if (args.data['duration'] == 0 && args.column.field === 'duration' )
{
            args.cell.style.background= '#336c12';
        } else if (args.data['duration'] < 3 && args.column.field ===
'duration') {
            args.cell.style.background= '#7b2b1d';
        }
      }
  },
  provide: {
      treegrid: [ Page, Toolbar, ExcelExport ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/excel/default-cs2" %}

Theme

The excel export provides an option to include theme for exported excel document.

To apply theme in exported Excel, define the theme in exportProperties.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowExcelExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
```

```
import { TreeGridPlugin, Page, Toolbar, ExcelExport } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['ExcelExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'Excel Export') {
          let excelExportProperties = {
            theme:
              {
                header: { fontName: 'Segoe UI', fontColor: '#666666' },
                record: { fontName: 'Segoe UI', fontColor: '#666666' },
                caption: { fontName: 'Segoe UI', fontColor: '#666666' }
              }
          }
          this.$refs.treegrid.excelExport(excelExportProperties);
        }
    }
  },
  provide: {
      treegrid: [ Page, Toolbar, ExcelExport ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/excel/default-cs3" %}

By default, material theme is applied to exported excel document.

Adding header and footer in Vue Treegrid component

The excel export provides an option to include header and footer content for exported excel document.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data' height='220'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowExcelExport='true'
:toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
```

```
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, ExcelExport } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      toolbarOptions: ['ExcelExport'],
      pageSettings: { pageSize: 7 }
    };
  },
  methods: {
      toolbarClick(args) {
        if (args['item'].text === 'Excel Export') {
          let excelExportProperties = {
            header: {
                headerRows: 7,
                rows: [
                    { cells: [{ colSpan: 4, value: "Northwind Traders", style: { fontColor: '#C67878', fontSize: 20, hAlign: 'Center', bold: true, } }] },
                    { cells: [{ colSpan: 4, value: "2501 Aerial Center Parkway", style: { fontColor: '#C67878', fontSize: 15, hAlign: 'Center', bold: true, } }] },
                    { cells: [{ colSpan: 4, value: "Suite 200 Morrisville, NC 27560 USA", style: { fontColor: '#C67878', fontSize: 15, hAlign: 'Center', bold: true, } }] },
                    { cells: [{ colSpan: 4, value: "Tel +1 888.936.8638 Fax +1 919.573.0306", style: { fontColor: '#C67878', fontSize: 15, hAlign: 'Center', bold: true, } }] },
                    { cells: [{ colSpan: 4, hyperlink: { target: 'https://www.northwind.com/', displayText: 'www.northwind.com' }, style: { hAlign: 'Center' } }] },
                    { cells: [{ colSpan: 4, hyperlink: { target: 'mailto:support@northwind.com' }, style: { hAlign: 'Center' } }] },
                ]
            },
            footer: {
                footerRows: 4,
                rows: [
                    { cells: [{ colSpan: 4, value: "Thank you for your business!", style: { hAlign: 'Center', bold: true } }] },
                    { cells: [{ colSpan: 4, value: "!Visit Again!", style: { hAlign: 'Center', bold: true } }] }
                ]
            },
          }
          this.$refs.treegrid.excelExport(excelExportProperties);
        }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  },
  provide: {
      treegrid: [ Page, Toolbar, ExcelExport ]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/excel/default-cs1" %}

Exporting tree grid in Server in Vue Tree Grid Component

The Tree Grid have an option to export the data to Excel in server side using tree grid server export library.

Server dependencies

The Server side export functionality is shipped in the Syncfusion.EJ2.TreeGridExport package, which is available in Essential Studio and nuget.org.The following list of dependencies is required for tree grid server side Excel exporting action.

- Syncfusion.EJ2
- Syncfusion.EJ2.TreeGridExport

Server configuration

The following code snippet shows server configuration using ASP.NET Core Controller Action.

To Export the tree grid in server side, You need to call the serverExcelExport method for passing the tree grid properties to server exporting action.

`ts

public IActionResult ExcelExport(string treeGridModel)

{

if (treeGridModel == null)

{

return View();

}

TreeGridExcelExport exp = new TreeGridExcelExport();

Syncfusion.EJ2.TreeGrid.TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);

return exp.ExportToExcel<TreeData>(gridProperty, TreeData.GetDefaultData());

}

private Syncfusion.EJ2.TreeGrid.TreeGrid ConvertTreeGridObject(string gridProperty)

{

Syncfusion.EJ2.TreeGrid.TreeGrid TreeGridModel =
(Syncfusion.EJ2.TreeGrid.TreeGrid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
typeof(Syncfusion.EJ2.TreeGrid.TreeGrid));

Copyright © 2001 -2024 Syncfusion Inc.

```
TreeGridColumnModel cols =
(TreeGridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
typeof(TreeGridColumnModel));

TreeGridModel.Columns = cols.columns;

return TreeGridModel;

}

public class TreeGridColumnModel

{

public List<TreeGridColumn> columns { get; set; }

}
```

`ts

```
<template>

<div id="app">

<ejs-treegrid ref='treegrid' :dataSource='data' height='220' childMapping='subtasks'
:treeColumnIndex='1' :allowExcelExport='true' :toolbar='toolbarOptions' :toolbarClick='toolbarClick'>

<e-columns>

<e-column field='taskID' headerText='Task ID' width='90' textAlign='Right'></e-column>

<e-column field='taskName' headerText='Task Name' width='160'></e-column>

<e-column field='startDate' headerText='Start Date' width='90' format="yMd" textAlign='Right'></e-
column>

<e-column field='duration' headerText='Duration' width='80' textAlign='Right'></e-column>

</e-columns>

</ejs-treegrid>

</div>

</template>

<script>

import Vue from "vue";

import { TreeGridPlugin, Page, Toolbar, ExcelExport } from "@syncfusion/ej2-vue-treegrid";

import { sampleData } from "./datasource.js";

Vue.use(TreeGridPlugin);

export default {

data ()  {

return {
```

```
data: sampleData,
toolbarOptions: ['ExcelExport'],
};
},
methods: {
toolbarClick(args) {
if (args['item'].text === 'Excel Export') {
this.$refs.treegrid.serverExcelExport('Home/ExcelExport');
}
}
},
provide: {
treegrid: [ Page, Toolbar, ExcelExport ]
}
}
</script>
`
```

CSV Export in server side

You can export the tree grid to CSV format by using the serverCsvExport method which will pass the tree grid properties to server.

In the below demo, we have invoked the above method inside the toolbarClick event. In server side, we have deserialized the tree grid properties and passed to the ExportToCsv method which will export the properties to CSV format.

```
`ts
public IActionResult ServerSideExporting()
{
var order = TreeData.GetDefaultData();
ViewBag.dataSource = order;
return View();
}
public IActionResult CsvExport(string treeGridModel)
{
if (treeGridModel == null)
{
```

```
return View();
}
TreeGridExcelExport exp = new TreeGridExcelExport();
Syncfusion.EJ2.TreeGrid.TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
return exp.ExportToCsv<TreeData>(gridProperty, TreeData.GetDefaultData());
}
private Syncfusion.EJ2.TreeGrid.TreeGrid ConvertTreeGridObject(string gridProperty)
{
Syncfusion.EJ2.TreeGrid.TreeGrid TreeGridModel =
(Syncfusion.EJ2.TreeGrid.TreeGrid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
typeof(Syncfusion.EJ2.TreeGrid.TreeGrid));
TreeGridColumnModel cols =
(TreeGridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
typeof(TreeGridColumnModel));
TreeGridModel.Columns = cols.columns;
return TreeGridModel;
}
public class TreeGridColumnModel
{
public List<TreeGridColumn> columns { get; set; }
}
`
```

```ts
<template>
<div id="app">
<ejs-treegrid ref='treegrid' :dataSource='data' height='220' childMapping='subtasks'
:treeColumnIndex='1' :toolbar='toolbarOptions' :toolbarClick='toolbarClick'>
<e-columns>
<e-column field='taskID' headerText='Task ID' width='90' textAlign='Right'></e-column>
<e-column field='taskName' headerText='Task Name' width='160'></e-column>
<e-column field='startDate' headerText='Start Date' width='90' format="yMd" textAlign='Right'></e-
column>
<e-column field='duration' headerText='Duration' width='80' textAlign='Right'></e-column>
</e-columns>
</ejs-treegrid>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
data ()  {
return {
data: sampleData,
toolbarOptions: ['CsvExport'],
};
},
methods: {
toolbarClick(args) {
if (args['item'].text === 'Csv Export') {
this.$refs.treegrid.serverCsvExport('Home/CsvExport');
}
}
},
provide: {
treegrid: [ Page, Toolbar, CsvExport ]
}
}
</script>
`
```

Rotate a header text to a certain degree in the exported grid on the server side

The TreeGrid has support to customize the column header styles such as changing text orientation, the font color, and so on in the exported Excel file. To achieve this requirement, use the ExcelHeaderCellRendering event of the tree grid.

The ExcelHeaderCellRendering will be triggered when creating a column header for the excel document to be exported in the server side. Customize the column header in this event.

Copyright © 2001 -2024 Syncfusion Inc.

In the following demo, using the HeaderCellRotate method of the TreeGridExcelExport class in the ExcelHeaderCellRendering event, you can rotate the header text of the column header in the excel exported document.

```ts
public IActionResult ExcelExport(string treeGridModel)
{
if (treeGridModel == null)
{
return View();
}
TreeGridExcelExport exp = new TreeGridExcelExport();
Syncfusion.EJ2.TreeGrid.TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
gridProperty.ExcelHeaderCellRendering = ExcelHeaderQueryCellInfo;
return (IActionResult)exp.ExcelExport<TreeGridItems>(gridProperty, TreeGridItems.GetDefaultData());
}
private void ExcelHeaderQueryCellInfo(object excel)
{
Syncfusion.EJ2.TreeGridExport.ExcelHeaderCellEventArgs name =
(Syncfusion.EJ2.TreeGridExport.ExcelHeaderCellEventArgs)excel;
List<string> headerValues = new List<string>();
headerValues.Add(name.Column.HeaderText);
var longestString = headerValues.Where(s => s.Length == headerValues.Max(m => m.Length)).First();
TreeGridExcelExport exp = new TreeGridExcelExport();
var size = exp.ExcelTextSize(name.Style.Font.FontName, (float)name.Style.Font.Size, longestString);
name.Cell.RowHeight = size.Width;
exp.HeaderCellRotate(name, 45); // Give the rotate degree value by the user.
name.Style.Borders.LineStyle = Syncfusion.XlsIO.ExcelLineStyle.None;
}
```

Global local in Vue Treegrid component

Localization

The Localization library allows you to localize default text content of the TreeGrid. The treegrid component has static text on some features (like toolbar area text, filter menu text, pager information text, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the

locale value and translation object.

Copyright © 2001 -2024 Syncfusion Inc.

The following list of properties and its values are used in the treegrid.

Locale keywords |Text

EmptyRecord | No records to display

True | true

False | false

ExpandAll | Expand All

CollapseAll | Collapse All

RowIndent | Indent

RowOutdent | Outdent

InvalidFilterMessage | Invalid Filter Data

FilterbarTitle | \s filter bar cell

Add | Add

Edit| Edit

Cancel| Cancel

Update| Update

Delete | Delete

Print | Print

Pdfexport | PDF Export

Excelexport | Excel Export

Wordexport | Word Export

Csvexport | CSV Export

Search | Search

Save | Save

EditOperationAlert | No records selected for edit operation

DeleteOperationAlert | No records selected for delete operation

SaveButton | Save

OKButton | OK

CancelButton | Cancel

EditFormTitle | Details of

AddFormTitle | Add New Record

ConfirmDelete | Are you sure you want to Delete Record?

SearchColumns | search columns

Matchs | No Matches Found

Copyright © 2001 -2024 Syncfusion Inc.

FilterButton | Filter

ClearButton | Clear

StartsWith | Starts With

EndsWith | Ends With

Contains | Contains

Equal | Equal

NotEqual | Not Equal

LessThan | Less Than

LessThanOrEqual | Less Than Or Equal

GreaterThan | Greater Than

GreaterThanOrEqual | Greater Than Or Equal

ChooseDate | Choose a Date

EnterValue | Enter the value

autoFitAll | Auto Fit all columns

autoFit | Auto Fit this column

Export | Export

FirstPage | First Page

LastPage | Last Page

PreviousPage | Previous Page

NextPage | Next Page

SortAscending | Sort Ascending

SortDescending | Sort Descending

EditRecord | Edit Record

DeleteRecord | Delete Record

Above | Above

Below | Below

AddRow | Add Row

FilterMenu | Filter

SelectAll | Select All

Blanks | Blanks

FilterTrue | True

FilterFalse | False

NoResult | No Matches Found

Copyright © 2001 -2024 Syncfusion Inc.

ClearFilter | Clear Filter

NumberFilter | Number Filters

TextFilter | Text Filters

DateFilter | Date Filters

MatchCase | Match Case

Between | Between

CustomFilter | Custom Filter

CustomFilterPlaceHolder | Enter the value

CustomFilterDatePlaceHolder | Choose a date

AND | AND

OR | OR

ShowRowsWhere | Show rows where:

currentPageInfo | {0} of {1} pages

totalItemsInfo | ({0} items)

firstPageTooltip | Go to first page

lastPageTooltip | Go to last page

nextPageTooltip | Go to next page

previousPageTooltip | Go to previous page

nextPagerTooltip | Go to next pager

previousPagerTooltip | Go to previous pager

pagerDropDown | Items per page

pagerAllDropDown | Items

All | All

Loading translations

To load translation object in an application, use load function of the L10n class.

The following example demonstrates the TreeGrid in Deutsch culture.

APP.VUE

```
<template>
<div id="app">
      <ejs-treegrid :dataSource='data' locale='de-DE'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowFiltering='true'
:filterSettings='filterSettings' :toolbar='toolbar'>
        <e-columns>
          <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
          <e-column field='taskName' headerText='Task Name' width='160'></e-
column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          <e-column field='startDate' headerText='Start Date' width='90'
format="yMd" textAlign='Right'></e-column>
          <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
        </e-columns>
      </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { L10n, setCulture } from '@syncfusion/ej2-base';
import { TreeGridPlugin, Page, Toolbar, Filter } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
setCulture('de-DE');
L10n.load({
    'de-DE': {
        'treegrid': {
            'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
            'Expand All': 'Alle erweitern',
            'Collapse All': 'Alles einklappen',
            "Print": "Drucken",
            "Pdfexport": "PDF-Export",
            "Excelexport": "Excel-Export",
            "Wordexport": "Word-Export",
            "FilterButton": "Filter",
            "ClearButton": "klar",
            "StartsWith": "Beginnt mit",
            "EndsWith": "Endet mit",
            "Contains": "Enthält",
            "Equal": "Gleich",
            "NotEqual": "Nicht gleich",
            "LessThan": "Weniger als",
            "LessThanOrEqual": "Weniger als oder gleich",
            "GreaterThan": "Größer als",
            "GreaterThanOrEqual": "Größer als oder gleich",
            "EnterValue": "Geben Sie den Wert ein",
            "FilterMenu": "Filter"
        },
        'pager': {
            'currentPageInfo': '{0} von {1} Seiten',
            'totalItemsInfo': '({0} Beiträge)',
            'firstPageTooltip': 'Zur ersten Seite',
            'lastPageTooltip': 'Zur letzten Seite',
            'nextPageTooltip': 'Zur nächsten Seite',
            'previousPageTooltip': 'Zurück zur letzten Seit',
            'nextPagerTooltip': 'Zum nächsten Pager',
            'previousPagerTooltip': 'Zum vorherigen Pager'
        },
        "dropdowns": {
            "noRecordsTemplate": "Keine Aufzeichnungen gefunden"
        },
        "datepicker": {
            "placeholder": "Wählen Sie ein Datum",
            "today": "heute"
        }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
});
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      pageSettings: { pageSize: 7 },
      filterSettings: { type: 'Menu' },
      toolbar: ['Print']
    };
  },
  provide: {
      treegrid: [ Page, Toolbar, Filter ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/local/default-cs1" %}

Internationalization

The Internationalization library is used to globalize number, date, and time values in treegrid component using format strings in the columns.format.

APP.VUE

```
<template>
<div id="app">
      <ejs-treegrid :dataSource='data' locale='de-DE'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowFiltering='true'
:filterSettings='filterSettings' :toolbar='toolbar'>
        <e-columns>
          <e-column field='orderID' headerText='Order ID' width='90'
textAlign='Right'></e-column>
          <e-column field='orderName' headerText='Order Name'
width='160'></e-column>
          <e-column field='price' headerText='Price' width='90'
:format='formatOptions' textAlign='Right'></e-column>
      </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { L10n, loadCldr, setCulture, setCurrencyCode } from
'@syncfusion/ej2-base';
import * as currencies from './currencies.json';
import * as cagregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
import * as numberingSystems from './numberingSystems.json';
import { TreeGridPlugin, Page, Toolbar, Filter } from "@syncfusion/ej2-vue-
treegrid";
import { formatData } from "./datasource.js";
setCulture('de-DE');
setCurrencyCode('EUR');
L10n.load({
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    'de-DE': {
        'treegrid': {
            'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
            'Expand All': 'Alle erweitern',
            'Collapse All': 'Alles einklappen',
            "Print": "Drucken",
            "Pdfexport": "PDF-Export",
            "Excelexport": "Excel-Export",
            "Wordexport": "Word-Export",
            "FilterButton": "Filter",
            "ClearButton": "klar",
            "StartsWith": "Beginnt mit",
            "EndsWith": "Endet mit",
            "Contains": "Enthält",
            "Equal": "Gleich",
            "NotEqual": "Nicht gleich",
            "LessThan": "Weniger als",
            "LessThanOrEqual": "Weniger als oder gleich",
            "GreaterThan": "Größer als",
            "GreaterThanOrEqual": "Größer als oder gleich",
            "EnterValue": "Geben Sie den Wert ein",
            "FilterMenu": "Filter"
        },
        'pager': {
            'currentPageInfo': '{0} von {1} Seiten',
            'totalItemsInfo': '({0} Beiträge)',
            'firstPageTooltip': 'Zur ersten Seite',
            'lastPageTooltip': 'Zur letzten Seite',
            'nextPageTooltip': 'Zur nächsten Seite',
            'previousPageTooltip': 'Zurück zur letzten Seit',
            'nextPagerTooltip': 'Zum nächsten Pager',
            'previousPagerTooltip': 'Zum vorherigen Pager'
        },
        "dropdowns": {
            "noRecordsTemplate": "Keine Aufzeichnungen gefunden"
        },
        "datepicker": {
            "placeholder": "Wählen Sie ein Datum",
            "today": "heute"
        }
    }
});
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: formatData,
      pageSettings: { pageSize: 7 },
      filterSettings: { type: 'Menu' },
      formatOptions: { format:'C2' , useGrouping: false,
minimumSignificantDigits:1, maximumSignificantDigits:3, currency:'EUR' },
      toolbar: ['Print']
    };
  },
  provide: {
      treegrid: [ Page, Toolbar, Filter ]
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/local/default-cs2" %}

* In the above sample, Price column is formatted by NumberFormatOptions.

* By default, locale value is en-US. If you want to change the en-US culture to a different culture, you have to change the locale accordingly.

Right to left (RTL)

RTL provides an option to switch the text direction and layout of the TreeGrid component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL Grid, set the enableRtl to true.

APP.VUE

```
<template>
<div id="app">
      <ejs-treegrid :dataSource='data' locale='ar-AE'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :allowFiltering='true'
:filterSettings='filterSettings' :toolbar='toolbar' :enableRtl='true'>
        <e-columns>
          <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
          <e-column field='taskName' headerText='Task Name' width='160'></e-
column>
          <e-column field='startDate' headerText='Start Date' width='90'
format="yMd" textAlign='Right'></e-column>
          <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
        </e-columns>
      </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { L10n, setCulture } from '@syncfusion/ej2-base';
import { TreeGridPlugin, Page, Toolbar, Filter } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
setCulture('ar-AE');
L10n.load({
    'ar-AE': {
        'treegrid': {
            "EmptyRecord": "لا سجلات لعرضها",
            "Print": "طباعة",
            "FilterButton": "مرشح",
            "ClearButton": "واضح",
            "StartsWith": "يبدأ ب",
            "EndsWith": "ينتهي مع",
            "Contains": "يحتوي على",
            "Equal": "مساو",
            "NotEqual": "غير متساوي",
            "LessThan": "أقل من",
```

```
            "LessThanOrEqual": "اصغر من او يساوي",
            "GreaterThan": "أكثر من",
            "GreaterThanOrEqual": "أكبر من أو يساوي",
            "ChooseDate": "اختر تاريخا",
            "EnterValue": "أدخل القيمة",
            "FilterMenu": "منقي"
        },
        'pager': {
            'currentPageInfo': '{0} صفحة} 1{ من',
            'totalItemsInfo': '({0} العناصر)',
            'firstPageTooltip': 'انتقل إلى الصفحة الأولى',
            'lastPageTooltip': 'انتقل إلى الصفحة الأخيرة',
            'nextPageTooltip': 'انتقل إلى الصفحة التالية',
            'previousPageTooltip': 'انتقل إلى الصفحة السابقة',
            'nextPagerTooltip': 'الذهاب إلى بيجر المقبل',
            'previousPagerTooltip': 'الذهاب إلى بيجر السابقة'
        },
        "dropdowns": {
            "noRecordsTemplate": "لا توجد سجلات"
        },
        "datepicker": {
            "placeholder": "اختر تاريخا",
            "today": "اليوم"
        }
    }
});
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      pageSettings: { pageSize: 7 },
      filterSettings: { type: 'Menu' },
      toolbar: ['Print']
    };
  },
  provide: {
      treegrid: [ Page, Toolbar, Filter ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/local/default-cs3" %}

Accessibility in Vue Treegrid component

The Tree Grid component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Tree Grid component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="intermediate"> |

Copyright © 2001 -2024 Syncfusion Inc.

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="intermediate"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The Tree Grid component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the Tree Grid component:

| Attributes | Purpose |

| --- | --- |

| role=treegrid | Used to convey a significant and contextual message to the user. |

| aria-selected | Accurately reflect the selection state, whether it's single-select or multi-select. |

| aria-expanded | It can be used to show whether a node is expanded or collapsed, making it easier for screen reader users to navigate and understand the hierarchy. |

Copyright © 2001 -2024 Syncfusion Inc.

| aria-sort | Indicate the current sorting order of a table column for users with disabilities, facilitating accessible data presentation and interaction. |

| aria-busy | Loading state to improve accessibility for users, particularly those relying on screen readers. |

| aria-invalid | To indicate whether the user's input in a form field is valid or invalid, aiding users, including those with disabilities, in understanding and correcting their input. |

| aria-grabbed | Provides accessibility information for users interacting with draggable elements |

| aria-owns | Establishing relationships between an element and the elements it owns or controls. |

| aria-label | Provides an accessible name for the close icon. |

Keyboard interaction

The Tree Grid component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Tree Grid component.

Interaction Keys |Description

PageDown |Goes to the next page.

PageUp |Goes to the previous page.

Ctrl + Alt +PageDown |Goes to the last page.

Ctrl + Alt + PageUp |Goes to the first page.

Alt + PageDown |Goes to the next page.

Alt + PageUp |Goes to the previous page.

Home |Goes to the first cell.

End |Goes to the last cell.

Ctrl + Home |Goes to the first row.

Ctrl + End |Goes to the last row.

DownArrow |Moves the cell focus downward.

UpArrow |Moves the cell focus upward.

LeftArrow |Moves the cell focus left side.

RightArrow |Moves the cell focus right side.

Shift + DownArrow |Extends the row/cell selection downwards.

Shift + UpArrow |Extends the row/cell selection upwards.

Shift + LeftArrow |Extends the cell selection to the left side.

Shift + RightArrow |Extends the cell selection to the right side.

Enter | Moves the row/cell selection downward. If current cell is in edit state, then completes the editing. If the current cell is a header then performs sorting.

Copyright © 2001 -2024 Syncfusion Inc.

Shift + Enter | Moves the row/cell selection upward. If the current cell is a header then clears sorting for the selected column.

Ctrl + Enter | If the current cell is a header then performs multi-sorting.

Tab | Moves the cell selection right side.

Shift + Tab | Moves the cell selection left side.

Esc |Deselects all the rows/cells.

Ctrl + A |Selects all the rows/cells.

UpArrow |Moves up a row/cell selection.

DownArrow |Moves down a row/cell selection.

RightArrow |Moves to the right cell selection.

LeftArrow |Moves to the left cell selection.

Ctrl + Shift + DownArrow |Expands the selected group.

Ctrl + DownArrow |Expands all the visible groups.

Ctrl + Shift + UpArrow |Collapses the selected group.

Ctrl + UpArrow |Collapses all the visible groups.

Ctrl + P |Prints the TreeGrid.

Ensuring accessibility

The Tree Grid component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Tree Grid component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Tree Grid component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/tree-grid.html" %}

See also

- Accessibility in Syncfusion Vue components

Clipboard in Vue Treegrid component

The clipboard provides an option to copy selected rows or cells data into the clipboard.

The following list of keyboard shortcuts is supported in the Tree Grid to copy selected rows or cells data into clipboard.

Interaction keys |Description

Ctrl + C |Copy selected rows or cells data into clipboard.

Ctrl + Shift + H |Copy selected rows or cells data with header into clipboard.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='treeGridData' childMapping='subtasks'
:treeColumnIndex='1' :allowPaging='true' :pageSettings='pageSettings'
height='260px' :selectionSettings='selectionOptions'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      treeGridData: sampleData,
      pageSettings: { pageSize: 10 },
      selectionOptions: { type: 'Multiple'}
    };
  },
  provide: {
      treegrid: [ Page ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/clipboard/default-cs1" %}

Copy to clipboard by external buttons

To copy selected rows or cells data into clipboard with help of external buttons, you need to invoke the copy method.

APP.VUE

```
<template>
<div id="app">
<ejs-button id='copy' @click.native='copy'>Copy</ejs-button>
        <ejs-button id='copyHeader'
@click.native='copyHeader'>CopyHeader</ejs-button>
        <ejs-treegrid ref=treegrid :dataSource='treeGridData'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' height='230px'
:selectionSettings='selectionOptions'>
            <e-columns>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
Vue.use(ButtonPlugin);
export default {
  data ()  {
    return {
      treeGridData: sampleData,
      pageSettings: { pageSize: 10 },
      selectionOptions: { type: 'Multiple'}
    };
  },
  provide: {
      treegrid: [ Page ]
    },
    methods: {
    copy: function() {
        this.$refs.treegrid.copy();
    },
    copyHeader: function() {
        this.$refs.treegrid.copy(true);
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/clipboard/default-cs2" %}

Copy Hierarchy Modes

Tree Grid provides support for a set of copy modes with copyHierarchyMode property. The below are the type of filter mode available in TreeGrid.

- **Parent** : This is the default copy hierarchy mode in Tree Grid. Clipboard value have the selected records with its parent records. If the selected records not have any parent record then the selected record will be in clipboard.
- **Child** : Clipboard value have the selected records with its child record. If the selected records do not have any child record then the selected records will be in clipboard.

Copyright © 2001 -2024 Syncfusion Inc.

- **Both** : Clipboard value have the selected records with its both parent and child record. If the selected records do not have any parent and child record then the selected records alone in clipboard.
- **None** : Only the Selected records will be in clipboard.

APP.VUE

```
<template>
    <div id="app">
    <div style="padding-top: 7px; float:left">Hierarchy Mode</div><div
style="padding-left: 10px; display: inline-block">
            <ejs-dropdownlist id='ddlelement' :dataSource='dropData'
value='Parent' :fields='fields' :change="onChange"></ejs-dropdownlist></div>
        <ejs-treegrid ref=treegrid :dataSource='treeGridData'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
copyHierarchyMode='Parent' :pageSettings='pageSettings' height='230px'
:selectionSettings='selectionOptions'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, CopyHierarchyType } from "@syncfusion/ej2-
vue-treegrid";
import { DropDownListPlugin, ChangeEventArgs} from "@syncfusion/ej2-vue-
dropdowns";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
Vue.use(DropDownListPlugin);
export default {
  data ()  {
    return {
      treeGridData: sampleData,
      pageSettings: { pageSize: 10 },
      selectionOptions: { type: 'Multiple'},
      fields:  { text: 'mode', value: 'id' },
      dropData: [
        { id: 'Parent', mode: 'Parent' },
        { id: 'Child', mode: 'Child' },
        { id: 'Both', mode: 'Both' },
        { id: 'None', mode: 'None' },
    ];
    };
  },
  provide: {
      treegrid: [ Page ]
```

```
    },
    methods: {
    onChange: function(e: ChangeEventArgs): void {
      let mode:CopyHierarchyType = e.value.toString();
      this.$refs.treegrid.ej2Instances.copyHierarchyMode = mode;
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/clipboard/default-cs3" %}

Limitations of Copy Functionality

- Only current view records will be available in copy clipboard.

AutoFill

AutoFill Feature allows you to copy the data of selected cells and paste it to another cells by just dragging the autofill icon of the selected cells up to required cells. This feature is enabled by defining enableAutoFill property as true.

APP.VUE

```
<template>
    <div id="app">
        <ejs-treegrid :dataSource='treeGridData' :enableAutoFill='true'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' height='220px'
:selectionSettings='selectionOptions' :editSettings='editSettings'
:toolbar='toolbar'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
:isPrimaryKey='true' width='90' textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Edit, Page, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      treeGridData: sampleData,
      pageSettings: { pageSize: 10 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Batch' },
      toolbar: ['Add', 'Update', 'Cancel'],
      selectionOptions: { type: 'Multiple', mode: 'Cell', cellSelectionMode:
'Box' }
    };
  },
  provide: {
      treegrid: [ Edit, Page, Toolbar ]
    }
  }
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/clipboard/default-cs4" %}

* If enableAutoFill is set to true, then the autofill icon will be displayed on cell selection to copy cells.

* It requires the selection mode to be **Cell**, cellSelectionMode to be **Box** and also Batch Editing should be enabled.

Limitations of AutoFill

- Since the string values are not parsed to number and date type, so when the selected string type cells are dragged to number type cells then it will display as **NaN**. For date type cells, when the selected string type cells are dragged to date type cells then it will display as an **empty cell**.
- Linear series and the sequential data generations are not supported in this autofill feature.

Paste

You can able to copy the content of a cell or a group of cells by selecting the cells and pressing Ctrl + C shortcut key and paste it to another set of cells by selecting the cells and pressing Ctrl + V shortcut key.

APP.VUE

```
<template>
    <div id="app">
        <ejs-treegrid :dataSource='treeGridData' childMapping='subtasks'
:treeColumnIndex='1' :allowPaging='true' :pageSettings='pageSettings'
height='220px' :selectionSettings='selectionOptions'
:editSettings='editSettings' :toolbar='toolbar'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
:isPrimaryKey='true' width='90' textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
```

```
import { TreeGridPlugin, Edit, Page, Toolbar } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      treeGridData: sampleData,
      pageSettings: { pageSize: 10 },
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Batch' },
      toolbar: ['Add', 'Update', 'Cancel'],
      selectionOptions: { type: 'Multiple', mode: 'Cell', cellSelectionMode:
'Box' }
    };
  },
  provide: {
      treegrid: [ Edit, Page, Toolbar ]
    }
  }
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/clipboard/default-cs5" %}

To perform paste functionality, it requires the selection mode to be **Cell**, cellSelectionMode to be **Box** and also Batch Editing should be enabled.

Limitations of Paste Functionality

- Since the string values are not parsed to number and date type, so when the copied string type cells are pasted to number type cells then it will display as **NaN**. For date type cells, when the copied string format cells are pasted to date type cells then it will display as an **empty cell**.

Context menu in Vue Treegrid component

The TreeGrid has options to show the context menu when right clicked on it. To enable this feature, you need to define either default or custom item in the contextMenuItems.

To use the context menu, inject the ContextMenu module in the treegrid.

The default items are in the following table.

Items| Description

AutoFit| Auto fit the current column.

AutoFitAll | Auto fit all columns.

Edit| Edit the current record.

Delete | Delete the current record.

Save | Save the edited record.

Cancel | Cancel the edited state.

PdfExport | Export the treegrid data as Pdf document.

Copyright © 2001 -2024 Syncfusion Inc.

ExcelExport | Export the treegrid data as Excel document.

CsvExport | Export the treegrid data as CSV document.

SortAscending | Sort the current column in ascending order.

SortDescending | Sort the current column in descending order.

FirstPage | Go to the first page.

PrevPage | Go to the previous page.

LastPage | Go to the last page.

NextPage | Go to the next page.

AddRow | Add new row to the treegrid.

Indent | Indents the record to one level of hierarchy.|

Outdent | Outdents the record to one level of hierarchy.|

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' :allowPaging='true' :pageSettings='pageSettings'
:allowSorting='true' :allowResizing='true' :editSettings='editSettings'
:allowPdfExport='true' :allowExcelExport='true'
:contextMenuItems="contextMenuItems">
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Resize, Sort, ContextMenu, Edit, Page, PdfExport,
ExcelExport, RowDD } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
      pageSettings: { pageSize: 7 },
      contextMenuItems: ['AutoFit', 'AutoFitAll', 'SortAscending',
'SortDescending',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        'Copy', 'Edit', 'Delete', 'Save', 'Cancel',
        'PdfExport', 'ExcelExport', 'CsvExport', 'FirstPage', 'PrevPage',
        'LastPage', 'NextPage', 'Group', 'Ungroup', 'Indent', 'Outdent'],
    };
  },
  provide: {
      treegrid: [ Resize, Sort, ContextMenu, Edit, Page, PdfExport,
ExcelExport ]
    }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/contextMenu/default-cs1" %}

Custom context menu items

The custom context menu items can be added by defining the contextMenuItems as a collection of contextMenuItemModel.

Actions for this customized items can be defined in the contextMenuClick event.

In the below sample, we have shown context menu item for parent rows to expand or collapse child rows.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1' :allowPaging='true'
:pageSettings='pageSettings' :contextMenuItems="contextMenuItems"
:contextMenuOpen='contextMenuOpen' :contextMenuClick='contextMenuClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, ContextMenu, Page, TreeGridComponent  } from
"@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
import { BeforeOpenCloseEventArgs } from '@syncfusion/ej2-vue-inputs';
import { MenuEventArgs } from '@syncfusion/ej2-vue-navigations';
import { getValue, isNullOrUndefined } from '@syncfusion/ej2-base';
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
```

```
      data: sampleData,
      pageSettings: { pageSize: 7 },
      contextMenuItems: [
          {text: 'Collapse the Row', target: '.e-content', id:
'collapserow'},
          {text: 'Expand the Row', target: '.e-content', id: 'expandrow'}
      ],
    };
  },
  provide: {
      treegrid: [ ContextMenu, Page ]
  },
  methods:{
      contextMenuOpen:function (arg: BeforeOpenCloseEventArgs) {
        let elem: Element = arg.event.target as Element;
        let uid = (elem.closest('.e-row') as
HTMLTableRowElement).getAttribute('data-uid');
        if (isNullOrUndefined(getValue('hasChildRecords',
this.$refs.treegrid.ej2Instances.grid.getRowObjectFromUID(uid).data))) {
          arg.cancel = true;
        } else {
          let flag: boolean = getValue('expanded',
this.$refs.treegrid.ej2Instances.grid.getRowObjectFromUID(uid).data);
          let val: string = flag ? 'none' : 'block';
          document.querySelectorAll('li#expandrow')[0].setAttribute('style',
'display: ' + val + ';');
          val = !flag ? 'none' : 'block';

document.querySelectorAll('li#collapserow')[0].setAttribute('style',
'display: ' + val + ';');
        }
      },
      contextMenuClick:function (args: MenuEventArgs) {
        let selectedRows = this.$refs.treegrid.getSelectedRows() as any;
        let selectedRecords = this.$refs.treegrid.getSelectedRecords() as
any;
        if (args.item.id === 'collapserow') {
          this.$refs.treegrid.collapseRow(selectedRows[0] as
HTMLTableRowElement, selectedRecords[0]);
        } else {
          this.$refs.treegrid.expandRow(selectedRows[0] as
HTMLTableRowElement, selectedRecords[0]);
        }
      }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/contextMenu/default-cs2" %}

Enable and disable context menu items dynamically

You can enable and disable the context menu items using the enableItems method in contextMenuOpen event.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<div id="app">
        <ejs-treegrid ref='treegrid' :dataSource='data'
childMapping='subtasks' :treeColumnIndex='1' :editSettings='editSettings'
:allowPaging='true' :pageSettings='pageSettings'
:contextMenuItems="contextMenuItems" :contextMenuOpen='contextMenuOpen'
:contextMenuClick='contextMenuClick'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
width='160'></e-column>
                <e-column field='startDate' headerText='Start Date'
width='90' format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, ContextMenu, Page, Edit, TreeGridComponent  } from
"@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
import { BeforeOpenCloseEventArgs } from '@syncfusion/ej2-vue-inputs';
import { MenuEventArgs } from '@syncfusion/ej2-vue-navigations';
import { getValue, isNullOrUndefined } from '@syncfusion/ej2-base';
Vue.use(TreeGridPlugin);
export default {
  data ()  {
    return {
      data: sampleData,
      editSettings: { allowDeleting: true, allowEditing: true, allowAdding:
true, mode: 'Row' },
      pageSettings: { pageSize: 7 },
      contextMenuItems: [
          {text: 'Edit Record', target: '.e-content', id: 'Edit_record'},
          { text: 'Delete Record', target: '.e-content', id: 'Delete_record'
},
      ],
    };
  },
  provide: {
      treegrid: [ ContextMenu, Page, Edit ]
  },
  methods:{
      contextMenuOpen:function (args: BeforeOpenCloseEventArgs) {
        if (args.rowInfo.rowData.hasChildRecords == true) {

this.$refs.treegrid.ej2Instances.grid.contextMenuModule.contextMenu.enableIt
ems(['Edit Record'],true);

this.$refs.treegrid.ej2Instances.grid.contextMenuModule.contextMenu.enableIt
ems(['Delete Record'],false);
        } else  {
```

```
this.$refs.treegrid.ej2Instances.grid.contextMenuModule.contextMenu.enableIt
ems(['Edit Record'],false);

this.$refs.treegrid.ej2Instances.grid.contextMenuModule.contextMenu.enableIt
ems(['Delete Record'],true);
        }
      },
      contextMenuClick:function (args: MenuEventArgs) {
        if(args.element.innerHTML == "Edit Record"){

this.$refs.treegrid.ej2Instances.grid.editModule.startEdit(args.rowInfo.row)
;
        }
        else if(args.element.innerHTML == "Delete Record"){

this.$refs.treegrid.ej2Instances.grid.editModule.deleteRecord(args.rowInfo.r
ow);
        }
      }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/contextMenu/default-cs3" %}

You can hide or show an item in context menu for specific area inside of treegrid by defining the target property.

Treegrid styling in Vue Treegrid component

To modify the TreeGrid appearance, you need to override the default CSS of treegrid. Please find the list of CSS classes and its corresponding section in treegrid. Also, you have an option to create your own custom theme for all the Vue controls using our Theme Studio.

Section|CSS class|Purpose of CSS class

Root|e-treegrid|This classes are in this root element (div) of the treegrid control.

Header|e-gridheader|This class is added in the root element of header element. In this class, You can override thin line between header and content of the treegrid.

||e-table|This class is added at 'table' of the treegrid header. This CSS class makes table width as 100 %.

||e-columnheader|This class is added at 'tr' of the treegrid header.

||e-headercell|This class is added in 'th' element of treegrid header. You can override background color of header and border color.

||e-headercelldiv|This class is add in div which present 'th' element in the header. we recommend you to use the e-headercelldiv to override skeleton of header.

Body|e-gridcontent|This class is added at root of body content. This is to override background color of the body.

||e-table|This class is added to table of content. This CSS class makes table width as 100 %.

Copyright © 2001 -2024 Syncfusion Inc.

||e-altrow|This class is added to alternate rows of treegrid. This is to override alternate row color of the treegrid.

||e-rowcell|This class is added to all cells in the treegrid. This is to override cells appearance and styling.

||e-groupcaption|This class is added to the 'td' of group caption which is to change the background color of caption cell.

||e-selectionbackground|This class is added to rowcell's of the treegrid. This is override selection.

||e-hover|This class adds to row of treegrid, while hovering the treegrid rows.

Pager|e-pager|This class is added to root element of the pager. This to change appearance of the background color and color of font.

||e-pagercontainer|This class is added to numeric items of the pager.

||e-parentmsgbar|This class is added to pager info of the pager.

Summary|e-gridfooter|This class is added to root of the summary div.

||e-summaryrow|This class is added to rows of treegrid summary.

||e-summarycell|This class is added to cells of summary row. This to override background color of summary.

How To

Refresh the data source in Vue Treegrid component

How to refresh the datasource

You can add/delete the datasource records through an external button. To reflect the datasource changes in Tree Grid, you need to assign the modified data to dataSource property.

Please follow the below steps to refresh the Tree Grid after datasource change.

Step 1:

Add/delete the datasource record by using the following code.

`ts

let customData = {

TaskID: 99,

TaskName: "New Data",

StartDate: new Date("02/03/2017"),

EndDate: new Date("02/03/2017"),

Duration: 10

};

// Added New Record.

this.data.unshift(customData);

// Delete record.

this.data.splice(selectedRow, 1);

Copyright © 2001 -2024 Syncfusion Inc.

`

Step 2:

Refresh the Tree Grid after the datasource change by assign the modified data to dataSource property.

`ts

this.data = [...this.data]; // Refresh the TreeGrid.

`

APP.VUE

```
<template>
    <div id="app">
        <div>
            <ejs-button  iconCss="e-icons e-play-icon" cssClass="e-flat"
:isPrimary="true" :isToggle="true" v-on:click.native="Add">Add</ejs-button>
            <ejs-button  iconCss="e-icons e-play-icon" cssClass="e-flat"
:isPrimary="true" :isToggle="true" v-on:click.native="Delete">Delete</ejs-
button>
             <ejs-treegrid :dataSource="data" :treeColumnIndex="1"
height='210px' idMapping= 'TaskID' parentIdMapping='parentID' ref="treegrid"
:editSettings="editSettings" :toolbar="toolbar">
              <e-columns>
               <e-column field="TaskID" headerText="Task ID"
:isPrimaryKey='true' width="70" textAlign="Right"></e-column>
               <e-column field="TaskName" headerText="Task Name" width="100"
></e-column>
               <e-column field="StartDate" headerText="Start Date"
format="yMd" width="100" textAlign="Right"></e-column>
               <e-column field="EndDate" headerText="Start Date"
format="yMd" width="100" textAlign="Right"></e-column>
               <e-column field="Duration" headerText="Duration" width="90"
textAlign="Right"></e-column>
              </e-columns>
            </ejs-treegrid>
          </div>
    </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, Edit } from "@syncfusion/ej2-vue-
treegrid";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { projectData } from './datasource.js';
Vue.use(TreeGridPlugin);
Vue.use(ButtonPlugin);
export default {
  data() {
    return {
      data: projectData,
      editSettings: { allowAdding:true, allowDeleting:true, allowEditing:
true },
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel']
    };
  },
  methods: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    Add() {
      let customData = {
        TaskID: 99,
        TaskName: "New Data",
        StartDate: new Date("02/03/2017"),
        EndDate: new Date("02/03/2017"),
        Duration: 10
      };
      this.data.unshift(customData);
      this.data = [...this.data];
    }
    Delete(){
     let selectedRow = this.$refs.treegrid.getSelectedRowIndexes().length;
     let selectedRowIndex = this.$refs.treegrid.getSelectedRowIndexes()[0];
       if (selectedRow > 0) {
         this.data.splice(selectedRowIndex, 1);
        }
       else {
          alert("No records selected for delete operation");
       }
       this.data = [...this.data];
      }
    }
  provide: {
    treegrid: [Page, Edit, Toolbar]
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs18" %}

Enable disable grid and its actions in Vue Treegrid component

You can enable/disable the Tree Grid and its actions by applying/removing corresponding CSS styles.

To enable/disable the Tree Grid and its actions, follow the given steps:

Step 1:

Create CSS class with custom style to override the default style of Tree Grid.

`

.disabletreegrid {

pointer-events: none;

opacity: 0.4;

}

.wrapper {

cursor: not-allowed;

}

`

Step 2:

Copyright © 2001 -2024 Syncfusion Inc.

Add/Remove the CSS class to the Tree Grid in the click event handler of Button.

`ts

btnClick(args) {

if (this.$refs.treegrid.$el.classList.contains('disablegrid')) {

this.$refs.treegrid.$el.classList.remove('disablegrid');

document.getElementById("TreeGridParent").classList.remove('wrapper');

}

else {

this.$refs.treegrid.$el.classList.add('disablegrid');

document.getElementById("TreeGridParent").classList.add('wrapper');

}

}

`

In the below demo, the button click will enable/disable the Tree Grid and its actions.

APP.VUE

```
<template>
    <div id="app">
        <div>
            <ejs-button  iconCss="e-icons e-play-icon" cssClass="e-flat"
:isPrimary="true" :isToggle="true" v-
on:click.native="btnClick">Enable/Disable Grid</ejs-button>
            <div id="TreeGridParent">
              <ejs-treegrid :dataSource="data" :treeColumnIndex="1"
height='210px' childMapping="subtasks" ref="treegrid"
:editSettings="editSettings" :toolbar="toolbar">
                <e-columns>
                 <e-column field="taskID" :isPrimaryKey="true"
headerText="Task ID" width="70" textAlign="Right"></e-column>
                 <e-column field="taskName" headerText="Task Name"
width="100"></e-column>
                 <e-column field="startDate" headerText="Start Date"
format="yMd" width="90" textAlign="Right"></e-column>
                 <e-column field="endDate" headerText="End Date" width="100"
format="yMd" textAlign="Right"></e-column>
                 <e-column field="duration" headerText="Duration" width="90"
textAlign="Right"></e-column>
                </e-columns>
              </ejs-treegrid>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, Edit } from "@syncfusion/ej2-vue-
treegrid";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { sampleData } from './datasource.js';
Vue.use(TreeGridPlugin);
Vue.use(ButtonPlugin);
export default {
  data() {
    return {
      data: sampleData,
      editSettings: { allowAdding:true, allowDeleting:true, allowEditing:
true },
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel']
    };
  },
  methods: {
    btnClick(args) {
      if (this.$refs.treegrid.$el.classList.contains('disabletreegrid')) {
          this.$refs.treegrid.$el.classList.remove('disabletreegrid');

document.getElementById("TreeGridParent").classList.remove('wrapper');
      }
      else {
          this.$refs.treegrid.$el.classList.add('disabletreegrid');

document.getElementById("TreeGridParent").classList.add('wrapper');
      }
    }
  },
  provide: {
    treegrid: [Page, Edit, Toolbar]
  }
}
</script>
<style>
  .disabletreegrid {
    pointer-events: none;
    opacity: 0.4;
  }
  .wrapper {
    cursor: not-allowed;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs12" %}

Change header text dynamically in Vue Treegrid component

You can change the column headerText dynamically through an external button.

Follow the given steps to change the header text dynamically:

Step 1:

Get the column object corresponding to the field name by using the getColumnByField method.

Then change the header Text value.

`ts

Copyright © 2001 -2024 Syncfusion Inc.

/ get the JSON object of the column corresponding to the field name */

let column = this.$refs.treegrid.getColumnByField("duration");

/ assign a new header text to the column */

column.headerText = "Changed Text";

`

Step 2:

To reflect the changes in the Tree Grid header, invoke the refreshColumns method.

`ts

this.$refs.treegrid.refreshColumns();

`

APP.VUE

```
<template>
<div id="app">
        <ejs-button @click.native='changeHeader'>Change HeaderText</ejs-
button>
        <ejs-treegrid :dataSource="data" :treeColumnIndex="1" height='280px'
childMapping="subtasks" ref="treegrid">
        <e-columns>
          <e-column field="taskID" headerText="Task ID" width="70"
textAlign="Right"></e-column>
          <e-column field="taskName" headerText="Task Name" width="100"></e-
column>
          <e-column field="startDate" headerText="Start Date" format="yMd"
width="90" textAlign="Right"></e-column>
          <e-column field="endDate" headerText="End Date" width="100"
format="yMd" textAlign="Right"></e-column>
          <e-column field="duration" headerText="Duration" width="90"
textAlign="Right" ></e-column>
          <e-column field="progress" headerText="Progress" width="90"
textAlign="Right" ></e-column>
        </e-columns>
       </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { ButtonPlugin } from "@syncfusion/ej2-vue-buttons";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
Vue.use(ButtonPlugin);
export default {
  data() {
    return {
      data: sampleData,
    };
  },
  methods: {
    changeHeader: function() {
```

```
      let column =  this.$refs.treegrid.getColumnByField("duration");
      column.headerText = "Changed Text";
      this.$refs.treegrid.refreshColumns();
    },
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs3" %}

Customize column styles in Vue Treegrid component

You can customise the appearance of header and content of the particular column using the customAttributes property.

To customize the Tree Grid column, follow the given steps:

Step 1:

Create a css class with custom style to override the default style for rowcell and headercell.

`

.e-treegrid .e-rowcell.customcss{

background-color: #ecedee;

font-family: 'Bell MT';

color: 'red';

font-size: '20px';

}

.e-treegrid .e-headercell.customcss{

background-color: #2382c3;

color: white;

font-family: 'Bell MT';

font-size: '20px';

}

`

Step 2:

Add the custom css class to particular column by using customAttributes property.

`ts

<e-column field="taskName" headerText="Task Name" :customAttributes="{class: 'customcss'}" width="100"></e-column>

`

APP.VUE

```
<template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<div id="app">
        <ejs-treegrid :dataSource='data' childMapping='subtasks'
:treeColumnIndex='1' height='280px'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID' width=70
textAlign='Right'></e-column>
                <e-column field='taskName' headerText='Task Name'
:customAttributes= "{class: 'customcss'}" width=100></e-column>
                <e-column field='startDate' headerText='Start Date' width=90
format="yMd" textAlign='Right'></e-column>
                <e-column field='duration' headerText='Duration' width=90
textAlign='Right'></e-column>
                <e-column field="progress" headerText="Progress" width="90"
textAlign="Right" ></e-column>
            </e-columns>
        </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
    };
  },
}
</script>
<style>
.e-treegrid .e-rowcell.customcss{
    background-color: #ecedee;
    font-family: 'Bell MT';
    color: 'red';
    font-size: '20px';
}
.e-treegrid .e-headercell.customcss{
    background-color: #2382c3;
    color: white;
    font-family: 'Bell MT';
    font-size: '20px';
}
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs6" %}

Custom tool tip for columns in Vue Treegrid component

You can achieve the custom tooltip(EJ2 Tooltip) for Tree Grid by using the queryCellInfo event.

Render the ToolTip component for the Tree Grid cells by using the following code in the queryCellInfo event.

`ts

Copyright © 2001 -2024 Syncfusion Inc.

tooltip (args) {

let tooltip = new Tooltip({

content: args.data[args.column.field].toString();

}, args.cell);

}

`

APP.VUE

```
<template>
<div id="app">
      <ejs-treegrid :dataSource="data" :treeColumnIndex="1" height='300px'
:queryCellInfo='queryCellInfoEvent' childMapping="subtasks" ref="treegrid">
        <e-columns>
          <e-column field="taskID" headerText="Task ID" width="70"
textAlign="Right"></e-column>
          <e-column field="taskName" headerText="Task Name" width="100"></e-
column>
          <e-column field="startDate" headerText="Start Date" format="yMd"
width="90" textAlign="Right"></e-column>
          <e-column field="endDate" headerText="Start Date" format="yMd"
width="90" textAlign="Right"></e-column>
          <e-column field="duration" headerText="Duration" width="90"
textAlign="Right"></e-column>
        </e-columns>
       </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { Tooltip } from "@syncfusion/ej2-popups";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
    };
  },
  methods: {
    queryCellInfoEvent: function (args) {
         let tooltip = new Tooltip({
          content: args.data[args.column.field].toString()
        }, args.cell);
      },
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs5" %}

Copyright © 2001 -2024 Syncfusion Inc.

Change orientation of header text in Vue Treegrid component

You can change the orientation of the header text by using the customAttributes property.

Ensure the following steps:

Step 1:

Create a css class with orientation style for treegrid header cell.

`

.orientationcss .e-headercelldiv {

transform: rotate(90deg);

}

`

Step 2:

Add the custom css class to particular column by using customAttributes property.

`ts

<e-column field='endDate' headerText='End Date' width='90' format="yMd" :customAttributes= "{class: 'orientationcss'}" textAlign='Center'></e-column>

`

Step 3:

Resize the header cell height in create event by using the following code.

`ts

setHeaderHeight: function(args) {

let textWidth = document.querySelector(".orientationcss > div").scrollWidth; //Obtain the width of the headerText content.

let headerCell = document.querySelectorAll(".e-headercell");

for (let i = 0; i < headerCell.length; i++) {

headerCell.item(i).style.height = textWidth + "px"; //Assign the obtained textWidth as the height of the headerCell.

}

}

`

APP.VUE

```
<template>
  <div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex="1" height='210px'
:created='setHeaderHeight' childMapping="subtasks" ref="treegrid">
        <e-columns>
          <e-column field="taskID" headerText="Task ID" width="70"
textAlign="Right"></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          <e-column field="taskName" headerText="Task Name" width="100"
></e-column>
          <e-column field="startDate" headerText="Start Date" format="yMd"
textAlign="Right" width="90"></e-column>
          <e-column field="endDate" headerText="End Date" width="90"
format="yMd" :customAttributes= "{class: 'orientationcss'}"
textAlign="Center"></e-column>
          <e-column field="duration" headerText="Duration" width="90"
textAlign="Right"></e-column>
          <e-column field="progress" headerText="progress" width="90"
textAlign="Right"></e-column>
        </e-columns>
       </ejs-treegrid>
  </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
    };
  },
  methods: {
    setHeaderHeight: function(args) {
      let textWidth = document.querySelector(".orientationcss >
div").scrollWidth; //Obtain the width of the headerText content.
      let headerCell = document.querySelectorAll(".e-headercell");
      for (let i = 0; i < headerCell.length; i++) {
        headerCell.item(i).style.height = textWidth + "px"; //Assign the
obtained textWidth as the height of the headerCell.
      }
    },
  }
}
</script>
<style>
   .orientationcss .e-headercelldiv {
    transform: rotate(90deg);
    padding-top:5px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs4" %}

Customize the icon for column menu in Vue Treegrid component

You can customize the column menu icon by overriding the default Tree Grid class **.e-icons.e-columnmenu** with a custom property **content** as mentioned below,

`

.e-treegrid .e-columnheader .e-icons.e-columnmenu::before {

content: "\e903";

}

`

In the below sample, Tree Grid is rendered with a customized column menu icon.

APP.VUE

```
<template>
  <div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex="1" height='300px'
:showColumnMenu='true' childMapping="subtasks" ref="treegrid">
        <e-columns>
          <e-column field="taskID" headerText="Task ID" width="70"
textAlign="Right"></e-column>
          <e-column field="taskName" headerText="Task Name" width="100"
></e-column>
          <e-column field="startDate" headerText="Start Date" format="yMd"
width="90" textAlign="Right"></e-column>
          <e-column field="endDate" headerText="End Date" width="90"
format="yMd" textAlign="Center"></e-column>
          <e-column field="duration" headerText="Duration" width="90"
textAlign="Right"></e-column>
          <e-column field="progress" headerText="progress" width="90"
textAlign="Right"></e-column>
        </e-columns>
       </ejs-treegrid>
  </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, ColumnMenu} from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
    };
  },
   provide: {
    treegrid: [ColumnMenu ]
  }
}
</script>
<style>
   .e-treegrid .e-columnheader .e-icons.e-columnmenu::before {
        content: "\e903";
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs9" %}

Copyright © 2001 -2024 Syncfusion Inc.

Customize the edit dialog in Vue Treegrid component

You can customize the appearance of the edit dialog in the actionComplete event based on **requestType** as **beginEdit** or **add**.

In the below example, we have changed the dialog's header text for editing and adding records.

APP.VUE

```
<template>
    <div id="app">
        <div>
            <ejs-treegrid :dataSource="data" :treeColumnIndex="1"
height='250px' :actionComplete='actionComplete' childMapping="subtasks"
ref="treegrid" :editSettings="editSettings" :toolbar="toolbar">
              <e-columns>
               <e-column field="taskID" :isPrimaryKey="true"
headerText="Task ID" width="70" textAlign="Right"></e-column>
               <e-column field="taskName" headerText="Task Name"
width="100"></e-column>
               <e-column field="startDate" headerText="Start Date"
format="yMd" width="90" textAlign="Right"></e-column>
               <e-column field="endDate" headerText="Start Date"
format="yMd" width="90" textAlign="Right"></e-column>
               <e-column field="duration" headerText="Duration" width="90"
textAlign="Right"></e-column>
              </e-columns>
             </ejs-treegrid>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, Edit } from "@syncfusion/ej2-vue-
treegrid";
import { sampleData } from './datasource.js';
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      editSettings: { allowAdding:true, allowDeleting:true, allowEditing:
true, mode: 'Dialog' },
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel']
    };
  },
  methods: {
    actionComplete(args) {
        if ((args.requestType === 'beginEdit' || args.requestType ===
'add')) {
            let dialog = args.dialog;
            dialog.height = 400;
            // change the header of the dialog
            dialog.header = args.requestType === 'beginEdit' ? 'Record of '
+ args.rowData['taskName'] : 'New Record';
        }
    }
```

```
  },
  provide: {
    treegrid: [Page, Edit, Toolbar]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs8" %}

Cascading drop down list with treegrid editing in Vue Treegrid component

You can achieve the Cascading DropDownList with Tree Grid Editing by using the Cell Edit Template feature.

In the below demo, Cascading DropDownList rendered for **Priority** and **Duration** column.

APP.VUE

```
<template>
  <div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex="1" height='250px'
idMapping= 'TaskID' parentIdMapping='parentID' ref="treegrid"
:editSettings="editSettings" :toolbar="toolbar">
        <e-columns>
          <e-column field="TaskID" headerText="Task ID" :isPrimaryKey='true'
width="70" textAlign="Right"></e-column>
          <e-column field="TaskName" headerText="Task Name" width="100"
></e-column>
          <e-column field="StartDate" headerText="Start Date" format="yMd"
editType= 'datepickeredit' width="100" textAlign="Right"></e-column>
          <e-column field="EndDate" headerText="End Date" width="100"
format="yMd" editType='datepickeredit' textAlign="Right"></e-column>
          <e-column field="Priority" headerText="Priority" width="90"
editType='dropdownedit' textAlign="Left" :edit="priorityParams"></e-column>
          <e-column field="Duration" headerText="Duration" width="90"
editType= 'dropdownedit' textAlign="Right" :edit="durationParams"></e-
column>
          <e-column field="Progress" headerText="Progress" width="90"
textAlign="Right"></e-column>
        </e-columns>
       </ejs-treegrid>
  </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, Edit } from "@syncfusion/ej2-vue-
treegrid";
import { projectData } from "./datasource.js";
import { DropDownList } from "@syncfusion/ej2-dropdowns";
import { Query } from '@syncfusion/ej2-data';
let priorityData = [
      { priorityName: 'Normal', priorityId: '1' },
      { priorityName: 'High', priorityId: '2' },
  ];
let durationData = [
    { durationValue: 2, priorityId: '1', durationId: 2 },
    { durationValue: 3, priorityId: '1', durationId: 3 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    { durationValue: 4, priorityId: '1', durationId: 4 },
    { durationValue: 11, priorityId: '2', durationId: 11 },
    { durationValue: 15, priorityId: '2', durationId: 15 },
    { durationValue: 20, priorityId: '2', durationId: 20 }
  ];
let priorityElem, durationElem, priorityObj, durationObj;
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: projectData,
      editSettings: { allowAdding:true, allowDeleting:true, allowEditing:
true, mode: 'Row' },
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
      priorityParams: {
        create: () => {
          priorityElem = document.createElement("input");
          return priorityElem;
        },
        read: () => {
          return priorityObj.text;
        },
        destroy: () => {
          priorityObj.destroy();
        },
        write: () => {
          priorityObj = new DropDownList({
            dataSource: priorityData,
            fields: { value: "priorityId", text: "priorityName" },
            floatLabelType: "Never",
            placeholder: "Select a Priority",
            change: () => {
              durationObj.enabled = true;
              let tempQuery = new Query().where(
                "priorityId",
                "equal",
                priorityObj.value
              );
              durationObj.query = tempQuery;
              durationObj.text = null;
              durationObj.dataBind();
            }
          });
          priorityObj.appendTo(priorityElem);
        }
      },
      durationParams: {
        create: () => {
          durationElem = document.createElement("input");
          return durationElem;
        },
        destroy: () => {
          durationObj.destroy();
        },
        read: () => {
          return durationObj.text;
        },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        write: () => {
          durationObj = new DropDownList({
            dataSource: durationData,
            enabled: false,
            fields: { value: "durationId", text: "durationValue" },
            floatLabelType: "Never",
            placeholder: "Select a duration"
          });
          durationObj.appendTo(durationElem);
        }
      },
    };
  },
  provide: {
    treegrid: [Page, Edit, Toolbar]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs2" %}

Provide custom data source and enabling filtering to drop down list in Vue Treegrid component

You can provide data source to the DropDownList by using the **params** of columns.edit property.

While setting new data source using edit params, you must specify a new **query** property for the DropDownList as follows,

`ts

priorityParams: {

params: {

allowFiltering: true,

dataSource: priorityData,

fields: { text: "priorityName", value: "priorityName" },

query: new Query(),

actionComplete: () => false

}

};

`

You can also enable filtering for the DropDownList by passing the allowFiltering as **true** to the edit params.

In the below demo, DropDownList is rendered with custom dataSource for the *Priority* column and enabled filtering to search DropDownList items.

APP.VUE

```
<template>
  <div id="app">
```

```
        <ejs-treegrid :dataSource="data" :treeColumnIndex="1" height='250px'
idMapping='TaskID' parentIdMapping='parentID' ref="treegrid"
:editSettings="editSettings" :toolbar="toolbar">
        <e-columns>
          <e-column field="TaskID" headerText="Task ID" :isPrimaryKey='true'
width="70" textAlign="Right"></e-column>
          <e-column field="TaskName" headerText="Task Name" width="100"
></e-column>
          <e-column field="StartDate" headerText="Start Date" format="yMd"
editType= 'datepickeredit' width="100" textAlign="Right"></e-column>
          <e-column field="EndDate" headerText="End Date" width="100"
format="yMd" editType='datepickeredit' textAlign="Right"></e-column>
          <e-column field="Priority" headerText="Priority" width="90"
editType= 'dropdownedit' textAlign="Left" :edit="priorityParams"></e-column>
        </e-columns>
       </ejs-treegrid>
  </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, Edit } from "@syncfusion/ej2-vue-
treegrid";
import { projectData } from "./datasource.js";
import { DropDownList } from "@syncfusion/ej2-dropdowns";
import { Query } from '@syncfusion/ej2-data';
let priorityData = [
      { priorityName: 'Normal', priorityId: '1' },
      { priorityName: 'High', priorityId: '2' },
      { priorityName: 'Low', priorityId: '3' },
      { priorityName: 'Critical', priorityId: '4' },
      { priorityName: 'Breaker', priorityId: '5' }
  ];
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: projectData,
      editSettings: { allowAdding:true, allowDeleting:true, allowEditing:
true, mode: 'Row' },
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
      priorityParams: {
        params: {
          allowFiltering: true,
          dataSource: priorityData,
          fields: { text: "priorityName", value: "priorityName" },
          query: new Query(),
          actionComplete: () => false
        }
      },
    };
  },
  provide: {
    treegrid: [Page, Edit, Toolbar]
  }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs17" %}

Restrict decimal points with treegrid editing in Vue Treegrid component

By default, the number of decimal places will be restricted to two in the NumericTextBox while editing the numeric column. We can restrict to type the decimal points in a NumericTextBox by using the **validateDecimalOnType** and **decimals** properties of NumericTextBox.

In the below demo, while editing the row we have restricted to type the decimal point value in the NumericTextBox of **Price** column.

APP.VUE

```
<template>
  <div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex="1" height='250px'
childMapping='subtasks' ref='treegrid' :editSettings='editSettings'
:toolbar="toolbar">
        <e-columns>
          <e-column field="orderID" headerText="Order ID"
:isPrimaryKey='true' width="70" textAlign="Right"></e-column>
          <e-column field="orderName" headerText="Order Name" width="100"
></e-column>
          <e-column field="orderDate" headerText="Order Date" format="yMd"
editType= 'datepickeredit' width="100" textAlign="Right"></e-column>
          <e-column field="units" headerText="Units" width="90"
textAlign="Right"  editType= "numericedit"></e-column>
          <e-column field="price" headerText="Price" width="100"
:edit="numericParams" editType="numericedit" width="90" format="c2"
textAlign="Right"></e-column>
        </e-columns>
       </ejs-treegrid>
  </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, Edit } from "@syncfusion/ej2-vue-
treegrid";
import { stackedData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: stackedData,
      editSettings: { allowAdding:true, allowDeleting:true, allowEditing:
true, mode: 'Row' },
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
      numericParams: {
        params: {
           decimals: 0,
           format: "N",
           validateDecimalOnType: true
        }
      },
    };
  },
```

```
  provide: {
    treegrid: [Page, Edit, Toolbar]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs19" %}

Exporting filtered data in Vue Treegrid component

You can export the filtered data by defining the resulted data in PdfExportProperties.dataSource before export.

In the below Pdf exporting demo, We have gotten the filtered data from the filteredResult of Tree Grid filterModule and then defines the resulted data in PdfExportProperties.dataSource and pass it to pdfExport method.

APP.VUE

```
<template>
  <div id="app">
        <ejs-treegrid :dataSource="data" :allowPaging='true'
:allowFiltering='true' :treeColumnIndex="1" idMapping='TaskID'
parentIdMapping='parentID' ref="treegrid" :toolbar="toolbarOptions"
:allowPdfExport='true' :pageSettings='pageSettings'
:toolbarClick='toolbarClick'>
        <e-columns>
          <e-column field="TaskID" headerText="Task ID" :isPrimaryKey='true'
width="70" textAlign="Right"></e-column>
          <e-column field="TaskName" headerText="Task Name" width="100"
></e-column>
          <e-column field="StartDate" headerText="Start Date" format="yMd"
editType= 'datepickeredit' width="100" textAlign="Right"></e-column>
          <e-column field="EndDate" headerText="End Date" width="100"
format="yMd" editType='datepickeredit' textAlign="Right"></e-column>
          <e-column field="Duration" headerText="Duration" width="90"
textAlign="Right"></e-column>
          <e-column field="Priority" headerText="Priority" width="90"
textAlign="Left"></e-column>
        </e-columns>
       </ejs-treegrid>
  </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Filter, Toolbar, PdfExport, ExcelExport} from
"@syncfusion/ej2-vue-treegrid";
import { projectData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: projectData,
      toolbarOptions: ['PdfExport'],
      pageSettings: { pageSize: 4, pageCount:4 },
    };
  },
```

```
  methods: {
    toolbarClick(args){
        if (args['item'].id.indexOf("pdfexport") != -1) {
           let pdfdata;
           pdfdata =
this.$refs.treegrid.ej2Instances.filterModule.filteredResult;
           let exportProperties = {
             dataSource: pdfdata,
           };
            this.$refs.treegrid.pdfExport(exportProperties);
       }
     },
  },
  provide: {
    treegrid: [Page, Filter, Toolbar, PdfExport, ExcelExport]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs13" %}

Exporting selected data in Vue Treegrid component

You can export the selected records data by passing it to PdfExportProperties.dataSource or ExcelExportProperties.dataSource property in the toolbarClick event.

In the below exporting demo, we can get the selected records using getSelectedRecords method and pass the selected data to pdfExport or excelExport methods using respective export properties..

APP.VUE

```
<template>
  <div id="app">
        <ejs-treegrid :dataSource="data" :allowPaging='true'
:treeColumnIndex="1" idMapping='TaskID' parentIdMapping='parentID'
ref="treegrid" :toolbar="toolbarOptions"  :allowPdfExport='true'
:allowExcelExport='true' :pageSettings='pageSettings'
:toolbarClick='toolbarClick' :selectionSettings='selectionOption'>
        <e-columns>
          <e-column field="TaskID" headerText="Task ID" :isPrimaryKey='true'
width="70" textAlign="Right"></e-column>
          <e-column field="TaskName" headerText="Task Name" width="100"
></e-column>
          <e-column field="StartDate" headerText="Start Date" format="yMd"
editType= 'datepickeredit' width="100" textAlign="Right"></e-column>
          <e-column field="EndDate" headerText="End Date" width="100"
format="yMd" editType='datepickeredit' textAlign="Right"></e-column>
          <e-column field="Duration" headerText="Duration" width="90"
textAlign="Right"></e-column>
          <e-column field="Priority" headerText="Priority" width="90"
textAlign="Left"></e-column>
        </e-columns>
       </ejs-treegrid>
  </div>
</template>
<script>
import Vue from "vue";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { TreeGridPlugin, Page, Toolbar, PdfExport, ExcelExport} from
"@syncfusion/ej2-vue-treegrid";
import { projectData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: projectData,
      toolbarOptions: ['PdfExport', 'ExcelExport'],
      pageSettings: { pageSize: 5, pageCount:5 },
      selectionOption: {type: 'Multiple'}
    };
  },
  methods: {
    toolbarClick(args){
         if (args['item'].id.indexOf("pdfexport") != -1) {
            let selectedRecords = this.$refs.treegrid.getSelectedRecords();
                  let exportProperties = {
                      dataSource: selectedRecords
                  };
               this.$refs.treegrid.pdfExport(exportProperties);
               }
             else if (args['item'].id.indexOf("excelexport") != -1) {
                  let selectedRecords =
this.$refs.treegrid.getSelectedRecords();
                  let exportProperties = {
                      dataSource: selectedRecords
                  };
               this.$refs.treegrid.excelExport(exportProperties);
            }
         },
  },
  provide: {
    treegrid: [Page, Toolbar, PdfExport, ExcelExport]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs14" %}

Show spinner on exporting in Vue Treegrid component

You can show/ hide spinner component while exporting the Tree Grid using showSpinner/ hideSpinner methods. You can use toolbarClick event to show spinner before exporting and hide a spinner in the pdfExportComplete or excelExportComplete event after the exporting.

In the toolbarClick event, based on the parameter **args.item.text** as **PDF Export** or **Excel Export** we can call the showSpinner method from Tree Grid instance.

In the pdfExportComplete or excelExportComplete event, We can call the hideSpinner method.

In the below demo, we have rendered the default spinner component when exporting the Tree Grid.

APP.VUE

```
<template>
  <div id="app">
```

```
        <ejs-treegrid :dataSource="data" :allowPaging='true'
:treeColumnIndex="1" idMapping='TaskID' parentIdMapping='parentID'
ref="treegrid" :toolbar="toolbarOptions"  :allowPdfExport='true'
:allowExcelExport='true' :pageSettings='pageSettings'
:toolbarClick='toolbarClick' :excelExportComplete='excelExportComplete'
:pdfExportComplete='pdfExportComplete'>
        <e-columns>
          <e-column field="TaskID" headerText="Task ID" :isPrimaryKey='true'
width="70" textAlign="Right"></e-column>
          <e-column field="TaskName" headerText="Task Name" width="100"
></e-column>
          <e-column field="StartDate" headerText="Start Date" format="yMd"
editType= 'datepickeredit' width="100" textAlign="Right"></e-column>
          <e-column field="EndDate" headerText="End Date" width="100"
format="yMd" editType='datepickeredit' textAlign="Right"></e-column>
          <e-column field="Duration" headerText="Duration" width="90"
textAlign="Right"></e-column>
          <e-column field="Priority" headerText="Priority" width="90"
textAlign="Left"></e-column>
        </e-columns>
       </ejs-treegrid>
  </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport, ExcelExport} from
"@syncfusion/ej2-vue-treegrid";
import { projectData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: projectData,
      toolbarOptions: ['PdfExport', 'ExcelExport'],
      pageSettings: { pageSize: 5, pageCount:5 },
      queryClone: ""
    };
  },
  methods: {
    toolbarClick(args){
        if (args.item.text === 'PDF Export') {
            this.$refs.treegrid.showSpinner();
            this.$refs.treegrid.pdfExport();
        }
        else if (args.item.text === 'Excel Export') {
            this.$refs.treegrid.showSpinner();
            this.$refs.treegrid.excelExport();
        }
     },
    pdfExportComplete(args) {
        this.$refs.treegrid.hideSpinner();
     },
     excelExportComplete(args) {
        this.$refs.treegrid.hideSpinner();
     },
  },
  provide: {
```

```
    treegrid: [Page, Toolbar, PdfExport, ExcelExport]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs21" %}

Passing parameter to server exporting in Vue Treegrid component

You can pass the additional parameter in the query property by invoking addParams method. In the toolbarClick event, you can define params as key and value pair so it will receive at the server side when exporting.

In the below example, we have passed *recordcount* as *12* using addParams method.

APP.VUE

```
<template>
  <div id="app">
        <ejs-treegrid :dataSource="data" :allowPaging='true'
:treeColumnIndex="1" idMapping='TaskID' parentIdMapping='parentID'
ref="treegrid" :toolbar="toolbarOptions"  :allowPdfExport='true'
:allowExcelExport='true' :pageSettings='pageSettings'
:toolbarClick='toolbarClick' :excelExportComplete='excelExportComplete'
:pdfExportComplete='pdfExportComplete' >
        <e-columns>
          <e-column field="TaskID" headerText="Task ID" :isPrimaryKey='true'
width="70" textAlign="Right"></e-column>
          <e-column field="TaskName" headerText="Task Name" width="100"
></e-column>
          <e-column field="StartDate" headerText="Start Date" format="yMd"
editType= 'datepickeredit' width="100" textAlign="Right"></e-column>
          <e-column field="EndDate" headerText="End Date" width="100"
format="yMd" editType='datepickeredit' textAlign="Right"></e-column>
          <e-column field="Duration" headerText="Duration" width="90"
textAlign="Right"></e-column>
          <e-column field="Priority" headerText="Priority" width="90"
textAlign="Left"></e-column>
        </e-columns>
       </ejs-treegrid>
  </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, PdfExport, ExcelExport} from
"@syncfusion/ej2-vue-treegrid";
import { projectData } from "./datasource.js";
import { Query } from '@syncfusion/ej2-data';
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: projectData,
      toolbarOptions: ['PdfExport', 'ExcelExport'],
      pageSettings: { pageSize: 5, pageCount:5 },
      queryClone: ""
    };
```

```
  },
  methods: {
    toolbarClick(args){
        if (args.item.text === 'PDF Export') {
            this.queryClone = this.$refs.treegrid.query;
            this.$refs.treegrid.query = new Query().addParams("recordcount",
"12")
            this.$refs.treegrid.pdfExport();
        }
        else if (args.item.text === 'Excel Export') {
            this.queryClone = this.$refs.treegrid.query;
            this.$refs.treegrid.query = new Query().addParams("recordcount",
"12")
            this.$refs.treegrid.excelExport();
        }
     },
     pdfExportComplete(args) {
        this.$refs.treegrid.query = this.queryClone();
     }
     excelExportComplete(args) {
        this.$refs.treegrid.query = this.queryClone();
     }
  },
  provide: {
    treegrid: [Page, Toolbar, PdfExport, ExcelExport]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs16" %}

Customize pager drop down in Vue Treegrid component

To customize default values of pager dropdown, you need to define pageSizes as array of strings.

APP.VUE

```
<template>
<div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex="1" height='250px'
childMapping="subtasks" ref="treegrid" :allowPaging="true"
        :pageSettings="pageSettings">
        <e-columns>
          <e-column field="taskID" headerText="Task ID" width="70"
textAlign='Right'></e-column>
          <e-column field="taskName" headerText="Task Name" width="100"
></e-column>
          <e-column field="startDate" headerText="Start Date" format="yMd"
width="90" textAlign='Right'></e-column>
          <e-column field="endDate" headerText="Start Date" format="yMd"
width="90" textAlign='Right'></e-column>
          <e-column field="duration" headerText="Duration" width="90"
textAlign='Right'></e-column>
          <e-column field="progress" headerText="Progress" width="90"
textAlign='Right'></e-column>
          <e-column field="priority" headerText="Priority" width="90"></e-
column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        </e-columns>
       </ejs-treegrid>
</div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { sampleData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: sampleData,
      pageSettings: {pageSizes: ['5','10','All']}
    };
  },
  provide: {
    treegrid: [Page]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs7" %}

Add parameter on filtering in Vue Treegrid component

You can customize the default settings of the components which are used in Menu filter by using params of filter property in column definition.

In the below sample, TaskID and Duration Columns are numeric columns, while open the filter dialog then you can see that NumericTextBox with spin button is displayed to change/set the filter value. Now using the params option we hide the spin button in NumericTextBox for TaskID Column.

APP.VUE

```
<template>
  <div id="app">
      <ejs-treegrid :dataSource="data" :treeColumnIndex="1" height='300px'
idMapping='TaskID' parentIdMapping='parentID' :allowFiltering='true'
:filterSettings='filterOptions' ref="treegrid">
        <e-columns>
          <e-column field="TaskID" headerText="Task ID" :isPrimaryKey='true'
width="70" :filter='filterParams' textAlign="Right"></e-column>
          <e-column field="TaskName" headerText="Task Name" width="100"
></e-column>
          <e-column field="StartDate" headerText="Start Date" format="yMd"
width="100" textAlign="Right"></e-column>
          <e-column field="EndDate" headerText="End Date" width="100"
format="yMd" textAlign="Right"></e-column>
          <e-column field="Duration" headerText="Duration" width="90"
:filter='filterParams' textAlign="Right"></e-column>
          <e-column field="Priority" headerText="Priority" width="90"
textAlign="Left"></e-column>
        </e-columns>
       </ejs-treegrid>
  </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Filter } from "@syncfusion/ej2-vue-treegrid";
import { projectData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: projectData,
      filterOptions: {
           type: 'Menu'
      },
      filterParams: { params: {showSpinButton: false}}
    };
  },
  provide: {
    treegrid: [Page, Filter]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs1" %}

Select treegrid rows based on certain condition in Vue Treegrid component

You can select the specific row in the Tree Grid based on a certain condition by using the selectRows method in the dataBound event of Tree Grid.

In the below demo, we have selected the Tree Grid rows only when *Duration* column value greater than *4*.

APP.VUE

```
<template>
  <div id="app">
      <ejs-treegrid :dataSource="data" :treeColumnIndex="1" height='300px'
idMapping='TaskID' parentIdMapping='parentID' :rowDataBound="rowDataBound"
:selectionSettings="selectionOptions"
        :dataBound="dataBound" ref="treegrid">
        <e-columns>
          <e-column field="TaskID" headerText="Task ID" :isPrimaryKey='true'
width="70" textAlign="Right"></e-column>
          <e-column field="TaskName" headerText="Task Name" width="100"
></e-column>
          <e-column field="StartDate" headerText="Start Date" format="yMd"
width="100" textAlign="Right"></e-column>
          <e-column field="EndDate" headerText="End Date" width="100"
format="yMd" textAlign="Right"></e-column>
          <e-column field="Duration" headerText="Duration" width="90"
textAlign="Right"></e-column>
          <e-column field="Priority" headerText="Priority" width="90"
textAlign="eft"></e-column>
        </e-columns>
       </ejs-treegrid>
  </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { projectData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      selIndex: [],
      selectionOptions: { type: "Multiple" },
      data: projectData,
    };
  },
  methods: {
    rowDataBound(args) {
       if (args.data["Duration"] > 3) {
        this.selIndex.push(parseInt(args.row.getAttribute("aria-
rowindex")));
      }
    },
    dataBound(args) {
      if (this.selIndex.length) {
        this.$refs.treegrid.selectRows(this.selIndex);
        this.selIndex = [];
      }
    }
  },
  provide: {
    treegrid: [Page]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs20" %}

Get row cell index in Vue Treegrid component

You can get the specific row and cell index of the Tree Grid by using rowSelected event of the treegrid. Here, we can get the row and cell index by using *aria-rowindex* (get row Index from *tr* element) and *aria-colindex* (column index from *td* element) attribute.

APP.VUE

```
<template>
  <div id="app">
      <ejs-treegrid :dataSource="data" :treeColumnIndex="1" height='300px'
idMapping='TaskID' parentIdMapping='parentID' :rowSelected='rowSelected'
ref="treegrid">
        <e-columns>
          <e-column field="TaskID" headerText="Task ID" :isPrimaryKey='true'
width="70" textAlign="Right"></e-column>
          <e-column field="TaskName" headerText="Task Name" width="100"
></e-column>
          <e-column field="StartDate" headerText="Start Date" format="yMd"
width="100" textAlign="Right"></e-column>
          <e-column field="EndDate" headerText="End Date" width="100"
format="yMd" textAlign="Right"></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          <e-column field="Duration" headerText="Duration" width="90"
textAlign="Right"></e-column>
          <e-column field="Priority" headerText="Priority" width="90"
textAlign="Left"></e-column>
        </e-columns>
       </ejs-treegrid>
  </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page } from "@syncfusion/ej2-vue-treegrid";
import { projectData } from "./datasource.js";
Vue.use(TreeGridPlugin);
export default {
  data() {
    return {
      data: projectData,
    };
  },
  methods: {
    rowSelected(args) {
        alert("row index: "+args.row.getAttribute('aria-rowindex'));
        alert("column index: "+args.target.closest('td').getAttribute('aria-
colindex'));
    }
  },
  provide: {
    treegrid: [Page]
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs15" %}

Display foreign key column in treegrid in Vue Treegrid component

Since Tree Grid Databinding concept is of hierarchy relationship, we do not provide in-built support for foreignkey datasource.

To display the foreignKey value at initial rendering, we can use the queryCellInfo event of the treegrid and also by using the editType and columns.edit properties of TreeGrid Column, we can render Dropdownlist with external or foreign dataSource.

APP.VUE

```
<template>
  <div id="app">
        <ejs-treegrid :dataSource="data" :treeColumnIndex="1" height='260px'
:allowSorting='true' :queryCellInfo='queryCellInfo' childMapping='Children'
:editSettings="editSettings" :toolbar="toolbar"  ref="treegrid">
        <e-columns>
         <e-column field="EmpID" headerText="Emp ID" :isPrimaryKey='true'
width="70" textAlign="Right"></e-column>
          <e-column field="Name" headerText="Employee Name" width="70" ></e-
column>
          <e-column field="Contact" headerText="Contact" width="90"
textAlign="Right"></e-column>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
          <e-column field="DOB" headerText="DOB" width="100" format="yMd"
editType='datepickeredit' textAlign="Right"></e-column>
          <e-column field="EmployeeID" headerText="EmployeeID" width="70"
editType= 'dropdownedit' :edit="employeeParams" ></e-column>
          <e-column field="Country" headerText="Country" width="90"
textAlign="Left"></e-column>
       </ejs-treegrid>
  </div>
</template>
<script>
import Vue from "vue";
import { TreeGridPlugin, Page, Toolbar, Edit } from "@syncfusion/ej2-vue-
treegrid";
import { textData, dropData } from "./datasource.js";
import { DropDownList } from "@syncfusion/ej2-dropdowns";
import { Query } from '@syncfusion/ej2-data';
Vue.use(TreeGridPlugin);
export default {
   data() {
    return {
      data: textData,
      editSettings: { allowAdding:true, allowDeleting:true, allowEditing:
true },
      toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
      employeeParams: {
        params: {
          dataSource: dropData,
          fields: { text: "EmployeeName", value: "EmployeeID"},
          query: new Query()
        }
      },
    };
  },
  methods: {
    queryCellInfo: function(args) {
      if (args.column.field === "EmployeeID") {
       for (var i = 0; i < dropData.length; i++) {
          let data = args.data;
          if ((data[args.column.field]).toString() ===
dropData[i]["EmployeeID"]) {
            args.cell.innerText = dropData[i]["EmployeeName"]; // assign the
foreignkey field value to the innertext
          }
        }
      }
    }
  },
  provide: {
    treegrid: [Page,Edit, Toolbar]
  }
</script>
```

{% previewsample "page.domainurl/code-snippet/treegrid/how-to/default-cs10" %}

Copyright © 2001 -2024 Syncfusion Inc.

TreeMap

Getting Started with the Vue TreeMap Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue TreeMap component

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following list of minimum dependencies are required to use the TreeMap:

```
|-- @syncfusion/ej2-treemap
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-compression
|-- @syncfusion/ej2-svg-base
```

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

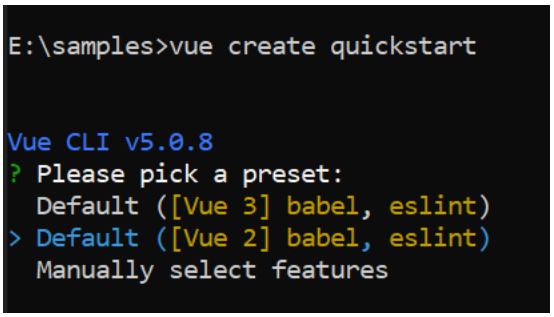
```bash
npm install -g @vue/cli
vue create quickstart
cd quickstart
npm run serve
```

or

```bash
yarn global add @vue/cli
vue create quickstart
cd quickstart
yarn run serve
```

When creating a new project, choose the option `Default ([Vue 2] babel, eslint)` from the menu.

Copyright © 2001 -2024 Syncfusion Inc.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Treemap component as an example. Install the @syncfusion/ej2-vue-treemap package by running the following command:

`bash

npm install @syncfusion/ej2-vue-treemap --save

`

or

`bash

yarn add @syncfusion/ej2-vue-treemap

`

Adding Syncfusion Vue TreeMap component

Follow the below steps to add the Vue TreeMap component:

1\. First, import and register the TreeMap component in the script section of the **src/App.vue** file.

~/SRC/APP.VUE

```
<script>
import { TreeMapComponent } from "@syncfusion/ej2-vue-treemap";
export default {
components: {
'ejs-treemap': TreeMapComponent
},
data: function() {
return {
dataSource: [
{ Title: 'State wise International Airport count in South America', State:
"Brazil", Count: 25 },
{ Title: 'State wise International Airport count in South America', State:
"Colombia", Count: 12 },
{ Title: 'State wise International Airport count in South America', State:
"Argentina", Count: 9 },
{ Title: 'State wise International Airport count in South America', State:
"Ecuador", Count: 7 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ Title: 'State wise International Airport count in South America', State:
"Chile", Count: 6 },
{ Title: 'State wise International Airport count in South America', State:
"Peru", Count: 3 },
{ Title: 'State wise International Airport count in South America', State:
"Venezuela", Count: 3 },
{ Title: 'State wise International Airport count in South America', State:
"Bolivia", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Paraguay", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Uruguay", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Falkland Islands",Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"French Guiana", Count:1 },
{ Title: 'State wise International Airport count in South America', State:
"Guyana", Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"Suriname", Count: 1 },
],
weightValuePath: 'Count',
leafItemSettings: {
labelPath: 'State',
}
}
}
}
</script>
```

2\. In the template section, define the TreeMap component.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-treemap :dataSource='dataSource' :weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
         <ejs-treemap :dataSource='dataSource'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import { TreeMapComponent } from '@syncfusion/ej2-vue-treemap';
export default {
  components: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    'ejs-treemap' : TreeMapComponent
  },
  data: function() {
    return {
        dataSource: [
            { Title: 'State wise International Airport count in South
America', State: "Brazil", Count: 25 },
            { Title: 'State wise International Airport count in South
America', State: "Colombia", Count: 12 },
            { Title: 'State wise International Airport count in South
America', State: "Argentina", Count: 9 },
            { Title: 'State wise International Airport count in South
America', State: "Ecuador", Count: 7 },
            { Title: 'State wise International Airport count in South
America', State: "Chile", Count: 6 },
            { Title: 'State wise International Airport count in South
America', State: "Peru", Count: 3 },
            { Title: 'State wise International Airport count in South
America', State: "Venezuela", Count: 3 },
            { Title: 'State wise International Airport count in South
America', State: "Bolivia", Count: 2 },
            { Title: 'State wise International Airport count in South
America', State: "Paraguay", Count: 2 },
            { Title: 'State wise International Airport count in South
America', State: "Uruguay", Count: 2 },
            { Title: 'State wise International Airport count in South
America', State: "Falkland Islands",Count: 1 },
            { Title: 'State wise International Airport count in South
America', State: "French Guiana", Count:1 },
            { Title: 'State wise International Airport count in South
America', State: "Guyana", Count: 1 },
            { Title: 'State wise International Airport count in South
America', State: "Suriname", Count: 1 },
        ],
        weightValuePath: 'Count',
        leafItemSettings: {
            labelPath: 'State',
        }
    }
  }
}
</script>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs60" %}

Module Injection

The tree map control is segregated into individual feature-wise modules. To use a particular feature, inject its feature module using the provide option on component creation. Find the modules available in tree map and their descriptions as follows.

- TreeMapHighlight - Inject this provider to use highlight feature.
- TreeMapSelection - Inject this provider to use selection feature.
- TreeMapLegend - Inject this provider to use legend feature.
- TreeMapTooltip - Inject this provider to use tooltip series.

In the current application, the above basic tree map is modified to visualize international airport count in South America.

In this demo, the tree map is just rendered with labels. For this, you need not to import any modules.

Render tree map

This section explains how to render the tree map with data source.

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap" :height='height' :dataSource='dataSource'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import { TreeMapComponent } from "@syncfusion/ej2-vue-treemap";
export default {
  components: {
    'ejs-treemap': TreeMapComponent
  },
  data: function() {
    return {
    height:'350px',
     dataSource: [
        { Title: 'State wise International Airport count in South America',
State: "Brazil", Count: 25 },
        { Title: 'State wise International Airport count in South America',
State: "Colombia", Count: 12 },
        { Title: 'State wise International Airport count in South America',
State: "Argentina", Count: 9 },
        { Title: 'State wise International Airport count in South America',
State: "Ecuador", Count: 7 },
        { Title: 'State wise International Airport count in South America',
State: "Chile", Count: 6 },
        { Title: 'State wise International Airport count in South America',
State: "Peru", Count: 3 },
        { Title: 'State wise International Airport count in South America',
State: "Venezuela", Count: 3 },
```

```
        { Title: 'State wise International Airport count in South America',
State: "Bolivia", Count: 2 },
        { Title: 'State wise International Airport count in South America',
State: "Paraguay", Count: 2 },
        { Title: 'State wise International Airport count in South America',
State: "Uruguay", Count: 2 },
        { Title: 'State wise International Airport count in South America',
State: "Falkland Islands",Count: 1 },
        { Title: 'State wise International Airport count in South America',
State: "French Guiana", Count:1 },
        { Title: 'State wise International Airport count in South America',
State: "Guyana", Count: 1 },
        { Title: 'State wise International Airport count in South America',
State: "Suriname", Count: 1 },
    ],
    weightValuePath: 'Count',
    leafItemSettings: {
        labelPath: 'State',
    }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs15" %}

Here, the tree map is created with data source and set with the [weightValuePath] as count. You can customize the leaf level tree map items using the [leafItemSettings]. The properties such as [fill], [border], and [labelPosition] can be changed using the [leafItemSettings].

Apply color mapping

The color mapping feature supports customization of item colors based on the underlying value of item received from bound data source. Specify the field name from the values that have to be compared for the item in the [equalColorValuePath] or [rangeColorValuePath] property.

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap" :height='height' :dataSource='dataSource'
:weightValuePath='weightValuePath'
:equalColorValuePath='equalColorValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import { TreeMapComponent } from "@syncfusion/ej2-vue-treemap";
export default {
  components: {
    'ejs-treemap': TreeMapComponent
  },
  data: function() {
    return {
     height: '350px',
        dataSource: [
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            { Title: 'State wise International Airport count in South
America', State: "Brazil", Count: 25 },
            { Title: 'State wise International Airport count in South
America', State: "Colombia", Count: 12 },
            { Title: 'State wise International Airport count in South
America', State: "Argentina", Count: 9 },
            { Title: 'State wise International Airport count in South
America', State: "Ecuador", Count: 7 },
            { Title: 'State wise International Airport count in South
America', State: "Chile", Count: 6 },
            { Title: 'State wise International Airport count in South
America', State: "Peru", Count: 3 },
            { Title: 'State wise International Airport count in South
America', State: "Venezuela", Count: 3 },
            { Title: 'State wise International Airport count in South
America', State: "Bolivia", Count: 2 },
            { Title: 'State wise International Airport count in South
America', State: "Paraguay", Count: 2 },
            { Title: 'State wise International Airport count in South
America', State: "Uruguay", Count: 2 },
            { Title: 'State wise International Airport count in South
America', State: "Falkland Islands", Count: 1 },
            { Title: 'State wise International Airport count in South
America', State: "French Guiana", Count: 1 },
            { Title: 'State wise International Airport count in South
America', State: "Guyana", Count: 1 },
            { Title: 'State wise International Airport count in South
America', State: "Suriname", Count: 1 },
        ],
        weightValuePath: 'Count',
        equalColorValuePath: 'Count',
        leafItemSettings: {
            labelPath: 'State',
            colorMapping: [
                {
                    value: 25,
                    color: '#634D6F'
                },
                {
                    value: 12,
                    color: '#B34D6D'
                },
                {
                    value: 9,
                    color: '#557C5C'
                },
                {
                    value: 7,
                    color: '#44537F'
                },
                {
                    value: 6,
                    color: '#637392'
                },
                {
                    value: 3,
                    color: '#7C754D'
```

```
                },
                {
                    value: 2,
                    color: '#2E7A64'
                },
                {
                    value: 1,
                    color: '#95659A'
                },
            ]
        }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs16" %}

Enable legend

You can show legend for the tree map by setting the [visible] property to true in [legendSettings] object and injecting the TreeMapLegend module using the provide option.

~/SRC/APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap" :dataSource='dataSource'
:legendSettings='legendSettings' :weightValuePath='weightValuePath'
:equalColorValuePath='equalColorValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import { TreeMapComponent, TreeMapLegend } from "@syncfusion/ej2-vue-
treemap";
export default {
  components: {
    'ejs-treemap': TreeMapComponent
  },
  data: function() {
    return {
     dataSource: [
            { Title: 'State wise International Airport count in South
America', State: "Brazil", Count: 25 },
            { Title: 'State wise International Airport count in South
America', State: "Colombia", Count: 12 },
            { Title: 'State wise International Airport count in South
America', State: "Argentina", Count: 9 },
            { Title: 'State wise International Airport count in South
America', State: "Ecuador", Count: 7 },
            { Title: 'State wise International Airport count in South
America', State: "Chile", Count: 6 },
            { Title: 'State wise International Airport count in South
America', State: "Peru", Count: 3 },
            { Title: 'State wise International Airport count in South
America', State: "Venezuela", Count: 3 },
```

```
            { Title: 'State wise International Airport count in South
America', State: "Bolivia", Count: 2 },
            { Title: 'State wise International Airport count in South
America', State: "Paraguay", Count: 2 },
            { Title: 'State wise International Airport count in South
America', State: "Uruguay", Count: 2 },
            { Title: 'State wise International Airport count in South
America', State: "Falkland Islands", Count: 1 },
            { Title: 'State wise International Airport count in South
America', State: "French Guiana", Count: 1 },
            { Title: 'State wise International Airport count in South
America', State: "Guyana", Count: 1 },
            { Title: 'State wise International Airport count in South
America', State: "Suriname", Count: 1 },
        ],
        legendSettings: {
            visible: true,
            position: 'Top',
            shape: 'Rectangle'
        },
        weightValuePath: 'Count',
        equalColorValuePath: 'Count',
        leafItemSettings: {
            labelPath: 'State',
            colorMapping: [
                {
                    value: 25,
                    color: '#634D6F'
                },
                {
                    value: 12,
                    color: '#B34D6D'
                },
                {
                    value: 9,
                    color: '#557C5C'
                },
                {
                    value: 7,
                    color: '#44537F'
                },
                {
                    value: 6,
                    color: '#637392'
                },
                {
                    value: 3,
                    color: '#7C754D'
                },
                {
                    value: 2,
                    color: '#2E7A64'
                },
                {
                    value: 1,
                    color: '#95659A'
                },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            ]
        }
    }
  },
provide:{
     treemap:[TreeMapLegend]
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs17" %}

Getting Started with the Vue TreeMap Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue TreeMap component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

Copyright © 2001 -2024 Syncfusion Inc.

This article uses the Vue TreeMap component as an example. To use the Vue TreeMap component in the project, the @syncfusion/ej2-vue-treemap package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-treemap --save

`

or

`bash

yarn add @syncfusion/ej2-vue-treemap

`

Add Syncfusion Vue component

Follow the below steps to add the Vue TreeMap component using Composition API or Options API:

1.First, import and register the TreeMap component and its child directives in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { TreeMapComponent as EjsTreemap, TreeMapLegend } from
"@syncfusion/ej2-vue-treemap";
import { TreeMap } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapLegend);
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { TreeMapComponent, TreeMapLegend } from "@syncfusion/ej2-vue-
treemap";
//Component registration
export default {
name: "App",
components: {
'ejs-treemap': TreeMapComponent
}
}
</script>
```

2.In the template section, define the TreeMap component with the dataSource, weightValuePath, leafItemSettings and other property definitions.

~/SRC/APP.VUE

```
<template>
<ejs-treemap id="treemap" :dataSource="dataSource"
:weightValuePath="weightValuePath" :leafItemSettings="leafItemSettings"
```

Copyright © 2001 -2024 Syncfusion Inc.

```
:equalColorValuePath="equalColorValuePath"
:legendSettings="legendSettings"></ejs-treemap>
</template>
```

3.In the script section, declare the values for the properties defined in the template section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const dataSource= [
{ Title: 'State wise International Airport count in South America', State:
"Brazil", Count: 25 },
{ Title: 'State wise International Airport count in South America', State:
"Colombia", Count: 12 },
{ Title: 'State wise International Airport count in South America', State:
"Argentina", Count: 9 },
{ Title: 'State wise International Airport count in South America', State:
"Ecuador", Count: 7 },
{ Title: 'State wise International Airport count in South America', State:
"Chile", Count: 6 },
{ Title: 'State wise International Airport count in South America', State:
"Peru", Count: 3 },
{ Title: 'State wise International Airport count in South America', State:
"Venezuela", Count: 3 },
{ Title: 'State wise International Airport count in South America', State:
"Bolivia", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Paraguay", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Uruguay", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Falkland Islands",Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"French Guiana", Count:1 },
{ Title: 'State wise International Airport count in South America', State:
"Guyana", Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"Suriname", Count: 1 }
];
const weightValuePath = 'Count';
const equalColorValuePath = "Count";
const leafItemSettings = {
labelPath: "State",
colorMapping: [
{ value: "25", color: "#634D6F" },
{ value: "12", color: "#B34D6D" },
{ value: "9", color: "#557C5C" },
{ value: "7", color: "#44537F" },
{ value: "6", color: "#637392" },
{ value: "3", color: "#7C754D" },
{ value: "2", color: "#2E7A64" },
{ value: "1", color: "#95659A" }
]
};
const legendSettings = {
visible: true
```

Copyright © 2001 -2024 Syncfusion Inc.

```
};
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
data() {
return {
dataSource: [
{ Title: 'State wise International Airport count in South America', State:
"Brazil", Count: 25 },
{ Title: 'State wise International Airport count in South America', State:
"Colombia", Count: 12 },
{ Title: 'State wise International Airport count in South America', State:
"Argentina", Count: 9 },
{ Title: 'State wise International Airport count in South America', State:
"Ecuador", Count: 7 },
{ Title: 'State wise International Airport count in South America', State:
"Chile", Count: 6 },
{ Title: 'State wise International Airport count in South America', State:
"Peru", Count: 3 },
{ Title: 'State wise International Airport count in South America', State:
"Venezuela", Count: 3 },
{ Title: 'State wise International Airport count in South America', State:
"Bolivia", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Paraguay", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Uruguay", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Falkland Islands",Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"French Guiana", Count:1 },
{ Title: 'State wise International Airport count in South America', State:
"Guyana", Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"Suriname", Count: 1 }
],
weightValuePath: 'Count',
equalColorValuePath: "Count",
leafItemSettings: {
labelPath: "State",
colorMapping: [
{ value: "25", color: "#634D6F" },
{ value: "12", color: "#B34D6D" },
{ value: "9", color: "#557C5C" },
{ value: "7", color: "#44537F" },
{ value: "6", color: "#637392" },
{ value: "3", color: "#7C754D" },
{ value: "2", color: "#2E7A64" },
{ value: "1", color: "#95659A" }
]
},
legendSettings: {
visible: true
}
```

```
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-treemap id="treemap" :dataSource="dataSource"
:weightValuePath="weightValuePath" :leafItemSettings="leafItemSettings"
:equalColorValuePath="equalColorValuePath"
:legendSettings="legendSettings"></ejs-treemap>
</template>
<script setup>
import { TreeMapComponent as EjsTreemap, TreeMapLegend } from
"@syncfusion/ej2-vue-treemap";
import { TreeMap } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapLegend);
const dataSource= [
{ Title: 'State wise International Airport count in South America', State:
"Brazil", Count: 25 },
{ Title: 'State wise International Airport count in South America', State:
"Colombia", Count: 12 },
{ Title: 'State wise International Airport count in South America', State:
"Argentina", Count: 9 },
{ Title: 'State wise International Airport count in South America', State:
"Ecuador", Count: 7 },
{ Title: 'State wise International Airport count in South America', State:
"Chile", Count: 6 },
{ Title: 'State wise International Airport count in South America', State:
"Peru", Count: 3 },
{ Title: 'State wise International Airport count in South America', State:
"Venezuela", Count: 3 },
{ Title: 'State wise International Airport count in South America', State:
"Bolivia", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Paraguay", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Uruguay", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Falkland Islands",Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"French Guiana", Count:1 },
{ Title: 'State wise International Airport count in South America', State:
"Guyana", Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"Suriname", Count: 1 }
];
const weightValuePath = 'Count';
const equalColorValuePath = "Count";
const leafItemSettings = {
labelPath: "State",
colorMapping: [
{ value: "25", color: "#634D6F" },
{ value: "12", color: "#B34D6D" },
```

```
{ value: "9", color: "#557C5C" },
{ value: "7", color: "#44537F" },
{ value: "6", color: "#637392" },
{ value: "3", color: "#7C754D" },
{ value: "2", color: "#2E7A64" },
{ value: "1", color: "#95659A" }
]
};
const legendSettings = {
visible: true
};
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-treemap id="treemap" :dataSource="dataSource"
:weightValuePath="weightValuePath" :leafItemSettings="leafItemSettings"
:equalColorValuePath="equalColorValuePath"
:legendSettings="legendSettings"></ejs-treemap>
</template>
<script>
import { TreeMapComponent, TreeMapLegend } from "@syncfusion/ej2-vue-
treemap";
// Component registration
export default {
name: "App",
// Declaring component and its directives
components: {
'ejs-treemap': TreeMapComponent
},
// Bound properties declarations
data() {
return {
dataSource: [
{ Title: 'State wise International Airport count in South America', State:
"Brazil", Count: 25 },
{ Title: 'State wise International Airport count in South America', State:
"Colombia", Count: 12 },
{ Title: 'State wise International Airport count in South America', State:
"Argentina", Count: 9 },
{ Title: 'State wise International Airport count in South America', State:
"Ecuador", Count: 7 },
{ Title: 'State wise International Airport count in South America', State:
"Chile", Count: 6 },
{ Title: 'State wise International Airport count in South America', State:
"Peru", Count: 3 },
{ Title: 'State wise International Airport count in South America', State:
"Venezuela", Count: 3 },
{ Title: 'State wise International Airport count in South America', State:
"Bolivia", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Paraguay", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Uruguay", Count: 2 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ Title: 'State wise International Airport count in South America', State:
"Falkland Islands",Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"French Guiana", Count:1 },
{ Title: 'State wise International Airport count in South America', State:
"Guyana", Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"Suriname", Count: 1 }
],
weightValuePath: 'Count',
equalColorValuePath: "Count",
leafItemSettings: {
labelPath: "State",
colorMapping: [
{ value: "25", color: "#634D6F" },
{ value: "12", color: "#B34D6D" },
{ value: "9", color: "#557C5C" },
{ value: "7", color: "#44537F" },
{ value: "6", color: "#637392" },
{ value: "3", color: "#7C754D" },
{ value: "2", color: "#2E7A64" },
{ value: "1", color: "#95659A" }
]
},
legendSettings: {
visible: true
}
};
},
provide: {
treemap:[TreeMapLegend]
}
};
</script>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

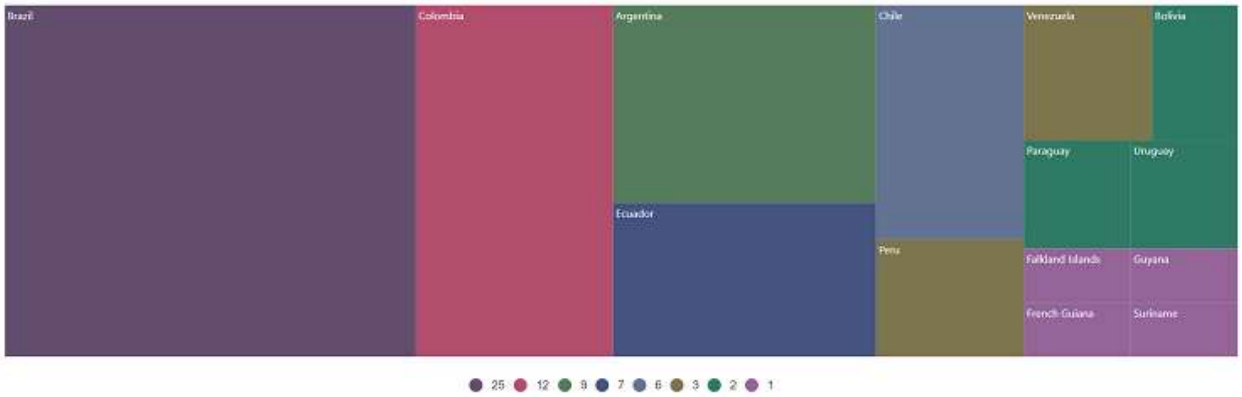
`bash

yarn run dev

`

The output will appear as follows:

Copyright © 2001 -2024 Syncfusion Inc.



Sample: vue3-treemap-getting-started.

For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Data binding in Vue Treemap component

The TreeMap control supports data binding using the dataSource property.

Populate data

The dataSource property accepts collection values as input. For example, a list of objects can be provided as input. Data can be given as either flat or hierarchical collection to the dataSource property.

<!-- markdownlint-disable MD036 -->

Flat collection

The following code shows, how to bind a flat collection as data source to the TreeMap control.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
        {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
```

```
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
    ],
    weightValuePath: 'GDP',
    leafItemSettings: {
        labelPath: 'State'
    }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs8" %}

Layout in Vue Treemap component

Determine the visual representation of nodes belonging to all the TreeMap levels using the layoutType property.

Types of layout

The available layout types are,

- Squarified
- SliceAndDiceVertical
- SliceAndDiceHorizontal
- SliceAndDiceAuto

Squarified

The Squarified layout displays the nested rectangles based on aspect ratio in the TreeMap. The rectangles will be split based on the height and width of the parent. The default rendering type of layout is Squarified.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
      return {
     dataSource: [
         {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
```

```
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
    ],
    weightValuePath: 'GDP',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs26" %}

SliceAndDiceVertical

The SliceAndDiceVertical layout creates rectangles with high aspect ratio and displays items in a vertically sorted order.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath' :layoutType='layoutType'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
         {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
    ],
    weightValuePath: 'GDP',
    layoutType : 'SliceAndDiceVertical',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs27" %}

Copyright © 2001 -2024 Syncfusion Inc.

SliceAndDiceHorizontal

The SliceAndDiceHorizontal layout creates rectangles with high aspect ratio and displays items in a horizontally sorted order.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath' :layoutType='layoutType'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
         {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
    ],
    layoutType : 'SliceAndDiceHorizontal',
    weightValuePath: 'GDP',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs28" %}

SliceAndDiceAuto

The SliceAndDiceAuto layout creates rectangles with high aspect ratio and display items sorted both horizontally and vertically.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath' :layoutType='layoutType'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data: function() {
      return {
     dataSource: [
         {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
    ],
    layoutType : 'SliceAndDiceAuto',
    weightValuePath: 'GDP',
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs29" %}

Leaf item in Vue Treemap component

A leaf item defines a visualized data element and does not contain child nodes but contains parent node if the levels are specified in the TreeMap.

Leaf label

Label is represented by item name or value. Label will be appeared by specifying the labelPath property and customize the label style using the labelStyle property.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap" :height='height' :dataSource='dataSource'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
    height:'350px',
     dataSource: [
        {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
    ],
    weightValuePath: 'GDP',
    leafItemSettings: {
            labelPath: 'State',
            labelStyle: {
                color: '#000000'
            },
            border: {
                color: '#000000',
                width: 0.5
            }
        },
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs30" %}

<!-- markdownlint-disable MD036 -->

Label position and format

Positioning the leaf item label using the labelPosition property and the text format can be customized by specifying data source properties name in the labelFormat property.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap" :height='height' :dataSource='dataSource'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
    height:'350px',
     dataSource: [
        {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    ],
    weightValuePath: 'GDP',
    leafItemSettings: {
            labelPath: 'State',
            labelPosition:'TopCenter',
            labelFormat: '${State}<br>$${GDP} Trillion<br>(${percentage}',
        },
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs31" %}

<!-- markdownlint-disable MD036 -->

Label template and position

Specifies the template of leaf item label and position of the template to be customized using labelTemplate and templatePosition properties.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap" :height='height' :dataSource='dataSource'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
    height:'350px',
     dataSource: [
       { Sport: "Swimming", Gold: 16, GameImage: 'Swimming.svg', ItemHeight:
"180px", ItemWidth: '180px' },
    { Sport: "Athletics", Gold: 13, GameImage: 'Athletics.svg', ItemHeight:
"70px", ItemWidth: '70px' },
    { Sport: "Gymnastics", Gold: 4, GameImage: 'Gymnastics.svg', ItemHeight:
"80px", ItemWidth: '80px' },
    { Sport: "Cycling", Gold: 2, GameImage: 'Cycling.svg', ItemHeight:
"50px", ItemWidth: '50px' },
    { Sport: "Wrestling", Gold: 2, GameImage: 'Wrestling.svg', ItemHeight:
"60px", ItemWidth: '50px' },
    { Sport: "Basketball", Gold: 2, GameImage: 'Basketball.svg', ItemHeight:
"50px", ItemWidth: '50px' },
    { Sport: "Boxing", Gold: 1, GameImage: 'Boxing.svg', ItemHeight: "40px",
ItemWidth: '30px' },
    { Sport: "Tennis", Gold: 1, GameImage: 'Tennis.svg', ItemHeight: "40px",
ItemWidth: '40px' },
    { Sport: "Judo", Gold: 1, GameImage: 'Judo.svg', ItemHeight: "40px",
ItemWidth: '40px' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    { Sport: "Rowing", Gold: 1, GameImage: 'Rowing.svg', ItemHeight: "40px",
ItemWidth: '40px' },
    { Sport: "Shooting", Gold: 1, GameImage: 'Shooting.svg', ItemHeight:
"40px", ItemWidth: '40px' },
    { Sport: "Triathlon", Gold: 1, GameImage: 'Triathlon.svg', ItemHeight:
"40px", ItemWidth: '40px' },
    { Sport: "Water polo", Gold: 1, GameImage: 'Water polo.svg', ItemHeight:
"40px", ItemWidth: '40px' }
    ],
    weightValuePath: 'Gold',
    leafItemSettings: {
            labelPath: 'Sport',
            fill: '#993399'
        },
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs32" %}

<!-- markdownlint-disable MD036 -->

Item gap

The gap property is used to separate an item from another item. Each item rectangle is split into equal space with specified gap.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap" :height='height' :dataSource='dataSource'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
    height:'350px',
     dataSource: [
        {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
    ],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    weightValuePath: 'GDP',
    leafItemSettings: {
        labelPath: 'State',
            gap:20
        },
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs33" %}

Levels in Vue Treemap component

TreeMap supports **n** number of levels and each level is separated by using the groupPath property.

<!-- markdownlint-disable MD036 -->

Group path

The groupPath property is used to separate each level of the TreeMap by specifying the property from the data source.

In the following example, three levels are added and each level is configured using the groupPath property.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap" :dataSource='dataSource'
:weightValuePath='weightValuePath' :levels='levels' :palette=
'palette'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }
    ],
    palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
"#009688"],
    weightValuePath: 'EmployeesCount',
    levels: [
        { groupPath: 'Country', border: { color: 'black', width: 0.5 } },
        { groupPath: 'JobDescription', border: { color: 'black', width: 0.5
} },
        { groupPath: 'JobGroup', border: { color: 'black', width: 0.5 } },
    ]
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs46" %}

<!-- markdownlint-disable MD036 -->

Group gap

The groupGap property is used to separate an item from each group or another item to differentiate the levels mentioned in the TreeMap.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath' :levels='levels' :palette=
'palette'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
```

```
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }
    ],
    palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
"#009688"],
    weightValuePath: 'EmployeesCount',
    levels: [
        { groupPath: 'Country', groupGap:10, border: { color: 'black',
width: 0.5 } },
        { groupPath: 'JobDescription', groupGap:10, border: { color:
'black', width: 0.5 } },
        { groupPath: 'JobGroup', groupGap:10, border: { color: 'black',
width: 0.5 } },
    ]
    }
  }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs47" %}

<!-- markdownlint-disable MD036 -->

Header format and Alignment

Customize header using the headerFormat property in which fields are mapping from the dataSource and align header using the headerAlignment property.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath' :levels='levels' :palette=
'palette'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }
    ],
    palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
"#009688"],
    weightValuePath: 'EmployeesCount',
    levels: [
        { groupPath: 'Country', headerFormat:'${Country}-${EmployeesCount}',
        headerAlignment:'Center', border: { color: 'black', width: 0.5 } },
        { groupPath: 'JobDescription',headerFormat:'${JobDescription}-
${EmployeesCount}', headerAlignment:'Far', border: { color: 'black', width:
0.5 } },
        { groupPath: 'JobGroup', headerAlignment:'Near',
        headerFormat:'${JobGroup}-${EmployeesCount}', border: { color:
'black', width: 0.5 } },
    ]
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs48" %}

Header height and style

Customize the font color, family, weight, opacity and size using the headerStyle. Based on the font settings, the header height is given using the headerHeight property in levels.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath' :levels='levels' :palette=
'palette'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }
    ],
    palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
"#009688"],
    weightValuePath: 'EmployeesCount',
    levels: [
        { groupPath: 'Country', headerHeight:35, headerStyle:{ size:'15px'
}, border: { color: 'black', width: 0.5 } },
        { groupPath: 'JobDescription',headerHeight:45, headerStyle:{
size:'15px' }, border: { color: 'black', width: 0.5 } },
        { groupPath: 'JobGroup',headerHeight:40, headerStyle:{ size:'15px'
}, border: { color: 'black', width: 0.5 } },
    ]
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs49" %}

Header template and position

The TreeMap header supports to customize header of each item using the headerTemplate property. It uses Essential JS2 Template engine to render the elements. You can position the template using the templatePosition property.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath' :levels='levels'
:leafItemSettings='leafItemSettings' :palette= 'palette'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }
    ],
    palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
"#009688"],
    weightValuePath: 'EmployeesCount',
    leafItemSettings:{
      showLabels:false
```

```
    },
    levels: [
        { groupPath: 'Country', headerFormat:' ',
headerTemplate:'<div>${Country}</div>', headerPosition:'Center', border: {
color: 'black', width: 0.5 } },
        { groupPath: 'JobDescription', headerFormat:' ',
headerTemplate:'<div>${JobDescription}</div>', headerPosition:'Center',
border: { color: 'black', width: 0.5 } },
        { groupPath: 'JobGroup', headerFormat:' ',
headerTemplate:'<div>${JobGroup}</div>', headerPosition:'Far', border: {
color: 'black', width: 0.5 } },
    ]
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs50" %}

Color mapping in Vue Treemap component

Color mapping is used to customize the color for each group or item based on the specified types. The following options are available to customize the group and leaf items in the TreeMap.

Range color mapping

Range color mapping is used to apply color to the items by giving specific ranges in the DataSource, and it should be specifying the data source properties to the rangeColorValuePath. The color mapping ranges to be specified in the from and to properties of the colorMapping.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:rangeColorValuePath='rangeColorValuePath'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapTooltip } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [{ fruit:'Apple', count:5000 },
                    { fruit:'Mango', count:3000 },
                    { fruit:'Orange', count:2300 },
                    { fruit:'Banana', count:500 },
                    { fruit:'Grape', count:4300 },
                    { fruit:'Papaya', count:1200 },
                    { fruit:'Melon', count:4500 }
    ],
    weightValuePath: 'count',
    rangeColorValuePath:'count',
    leafItemSettings: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        labelPath: 'fruit',
        colorMapping:[
            {
                from:500,
                to:3000,
                color:'orange'
            },
            {
                from:3000,
                to:5000,
                color:'green'
            }
        ]
     },
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs1" %}

Equal color mapping

Equal color mapping is used to fill colors to each item by specifying equal value present in the data source, that can be specified in the equalColorValuePath property.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:equalColorValuePath='equalColorValuePath'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [{ Car:'Mustang', Brand:'Ford', count:232 },
                    { Car:'EcoSport', Brand:'Ford', count:121 },
                    { Car:'Swift', Brand:'Maruti', count:143 },
                    { Car:'Baleno', Brand:'Maruti', count:454 },
                    { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
                    { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
                    { car:'RS7 Sportback', Brand:'Audi', count:523 }
    ],
    weightValuePath: 'count',
    equalColorValuePath:'Brand',
    leafItemSettings: {
        labelPath: 'Car',
        colorMapping:[
            {
```

```
                value:'Ford',
                color:'green'
            },
            {
                value:'Audi',
                color:'red'
            },
            {
                value:'Maruti',
                color:'orange'
            }
        ]
     },
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs2" %}

Desaturation color mapping

Desaturation color mapping is used to apply colors to the items based on the minOpacity and maxOpacity properties in the colorMapping.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:rangeColorValuePath='rangeColorValuePath'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
    height:'350px',
     dataSource: [
                        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
    weightValuePath: 'count',
    rangeColorValuePath:'count',
    leafItemSettings: {
        labelPath: 'fruit',
        colorMapping:[
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            {
                from:500,
                to:3000,
                minOpacity:0.2,
                maxOpacity:0.5,
                color:'orange'
            },
            {
                from:3000,
                to:5000,
                minOpacity:0.5,
                maxOpacity:0.8,
                color:'green'
            }
        ]
     }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs3" %}

Palette color mapping

The palette color mapping is used to fill the color to each group or leaf item by given colors in the palette property.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath' :palette='palette'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
    height:'350px',
     dataSource: [{ Car:'Mustang', Brand:'Ford', count:232 },
                    { Car:'EcoSport', Brand:'Ford', count:121 },
                    { Car:'Swift', Brand:'Maruti', count:143 },
                    { Car:'Baleno', Brand:'Maruti', count:454 },
                    { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
                    { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
                    { car:'RS7 Sportback', Brand:'Audi', count:523 }
                  ],
    palette:['red','green'],
    weightValuePath: 'count',
    leafItemSettings: {
        labelPath: 'Car'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
      }
    }
    }
  }
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs4" %}

Desaturation with multiple colors

Multiple colors are used as gradient effect to specific shapes based on the ranges in datasource. By using color property, you can set n number of colors.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:rangeColorValuePath='rangeColorValuePath'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
    height:'350px',
     dataSource: [
                        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
    weightValuePath: 'count',
    rangeColorValuePath:'count',
    leafItemSettings: {
        labelPath: 'fruit',
        colorMapping:[
            {
                from:500,
                to:2500,
                color:['orange','pink']
            },
            {
                from:3000,
                to:5000,
                color:['green','red','blue']
            }
        ]
     }
```

```
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs5" %}

Color for items excluded from color mapping

Get the excluded ranges from data source using the color mapping and apply the specific color to those items, without specifying the from and to properties.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:rangeColorValuePath='rangeColorValuePath'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
    height:'350px',
     dataSource: [
                        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
    weightValuePath: 'count',
    rangeColorValuePath:'count',
    leafItemSettings: {
        labelPath: 'fruit',
        colorMapping:[
            {
               from:500,
               to:2500,
               color:'orange'
           },
           {
               from:3000,
               to:4000,
               color:'green'
           },
           {
               color:'red'
           }
```

```
        ]
      }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs6" %}

Bind the colors to the items from data source

To set the color for each item from the data source, bind the data source property to the colorValuePath.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:colorValuePath='colorValuePath' :palette='palette'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
    height:'350px',
     dataSource: [
                        { fruit:'Apple', count:5000, color: 'red'},
                        { fruit:'Mango', count:3000, color:'blue'},
                        { fruit:'Orange', count:2300, color:'green' },
                        { fruit:'Banana', count:500, color:'yellow' },
                        { fruit:'Grape', count:4300, color:'orange' },
                        { fruit:'Papaya', count:1200, color:'pink' },
                        { fruit:'Melon', count:4500, color:'violet' }
    ],
    weightValuePath: 'count',
    colorValuePath:'color',
    palette:[],
    leafItemSettings: {
        labelPath: 'fruit',
     }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs7" %}

Copyright © 2001 -2024 Syncfusion Inc.

Data label in Vue Treemap component

Data Labels are used to identify the name of items or groups in the TreeMap component. Data Labels will be shown by specifying the data source properties in the labelPath of the leafItemSettings.

Format

Customize the labels for each item using the labelFormat property in the leafItemSettings.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
         { Car:'Mustang', Brand:'Ford', count:232 },
                    { Car:'EcoSport', Brand:'Ford', count:121 },
                    { Car:'Swift', Brand:'Maruti', count:143 },
                    { Car:'Baleno', Brand:'Maruti', count:454 },
                    { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
                    { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
                    { Car:'RS7 Sportback', Brand:'Audi', count:523 }
    ],
    weightValuePath: 'count',
    leafItemSettings: {
        labelPath: 'Car',
        labelFormat:'${Car}-${Brand}'
     }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs9" %}

Template

The template supports customizing labels of each leaf node using the labelTemplate property. It uses Essential JS2 template engine to render elements and the position of templates can be customize using the templatePosition property.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
let contentVue = Vue.component("contentTemplate", {
  template: '<div>{{data.Car}}-{{data.Brand}}</div>',
  data() {
    return {
      data: {}
    };
  }
});
let contentTemplate = function() {
  return { template: contentVue };
};
export default {
  data: function() {
    return {
     dataSource: [
         { Car:'Mustang', Brand:'Ford', count:232 },
                       { Car:'EcoSport', Brand:'Ford', count:121 },
                       { Car:'Swift', Brand:'Maruti', count:143 },
                       { Car:'Baleno', Brand:'Maruti', count:454 },
                       { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
                       { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
                       { Car:'RS7 Sportback', Brand:'Audi', count:523 }
    ],
    weightValuePath: 'count',
    leafItemSettings: {
       labelPath: 'Car',
       labelTemplate:contentTemplate
     }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs10" %}

InterSectAction

When the label size in each item exceeds the actual size, use the interSectAction property in the leafItemSettings to customise the labels.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
```

```
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
         { Car:'Mustang', Brand:'Ford', count:232 },
                       { Car:'EcoSport', Brand:'Ford', count:121 },
                       { Car:'Swift', Brand:'Maruti', count:143 },
                       { Car:'Baleno', Brand:'Maruti', count:454 },
                       { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
                       { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
                       { car:'RS7 Sportback', Brand:'Audi', count:523 }
    ],
    weightValuePath: 'count',
    leafItemSettings: {
        labelPath: 'Car',
        labelFormat:'${Car}-${Brand}',
        interSectAction:'WrapByWord'
     }
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs11" %}

Legend in Vue Treemap component

<!-- markdownlint-disable MD010 -->

Legend is used to provide valuable information for interpreting what the TreeMap displays. The legends can be represented in various colors, shapes or other identifiers based on the data.

Position and alignment

Legend position is used to place legend in various positions. Based on the legend position, the legend item will be aligned. For example, if the position is top or bottom, the legend items are placed by rows. If the position is left or right, the legend items are placed by columns.

The following options are available to customize the legend position:

- Top
- Bottom
- Left
- Right
- Float

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:legendSettings='legendSettings' :weightValuePath='weightValuePath'
:rangeColorValuePath= 'rangeColorValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
                        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
   weightValuePath: 'count',
    rangeColorValuePath:'count',
    legendSettings: {
        visible: true,
        position: 'Top'
    },
    leafItemSettings: {
        labelPath: 'fruit',
        colorMapping:[
            {
                from:500,
                to:3000,
                color:'orange'
            },
            {
                from:3000,
                to:5000,
                color:'green'
            }
        ]
     }
    }
  },
  provide:{
    treemap:[TreeMapLegend]
},
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs34" %}

Legend Alignment is used to align the legend items in specific location. The following options are available to customize the legend alignment:

- Near
- Center
- Far

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:rangeColorValuePath='rangeColorValuePath' :legendSettings='legendSettings'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
    height:'350px',
     dataSource: [
        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
   weightValuePath: 'count',
    rangeColorValuePath:'count',
    legendSettings: {
       visible: true,
       alignment:'Far',
    },
    leafItemSettings: {
        labelPath: 'fruit',
        colorMapping:[
            {
               from:500,
               to:3000,
               color:'orange'
           },
           {
               from:3000,
               to:5000,
               color:'green'
           }
        ]
     }
    }
  },
   provide:{
    treemap:[TreeMapLegend]
    },
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs35" %}

Legend mode

The TreeMap control supports two different types of legend rendering modes such as Default and Interactive.

<!-- markdownlint-disable MD036 -->

Default mode

In default mode, the legends have symbols with legend labels that are used to identify the items in the TreeMap.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:equalColorValuePath='equalColorValuePath' :legendSettings='legendSettings'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [{ Car:'Mustang', Brand:'Ford', count:232 },
                    { Car:'EcoSport', Brand:'Ford', count:121 },
                    { Car:'Swift', Brand:'Maruti', count:143 },
                    { Car:'Baleno', Brand:'Maruti', count:454 },
                    { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
                    { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
                    { Car:'RS7 Sportback', Brand:'Audi', count:523 }
    ],
   weightValuePath: 'count',
    equalColorValuePath:'Brand',
    legendSettings: {
        visible: true,
        position:'Top',
        border:{color:'black',width:2}
    },
    leafItemSettings: {
        labelPath: 'Car',
        colorMapping:[
            {
               value:'Ford',
               color:'green'
           },
           {
               value:'Audi',
               color:'red'
           },
           {
               value:'Maruti',
```

```
                color:'orange'
            }
        ]
      }
    }
  },
   provide:{
    treemap:[TreeMapLegend]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs36" %}

<!-- markdownlint-disable MD036 -->

Interactive mode

The legends can be made interactive with an arrow mark that indicates exact range color in the legend when the mouse hovers on the TreeMap item. Enable this option by setting the mode property in the legendSettings to **Interactive**.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:equalColorValuePath='equalColorValuePath' :legendSettings='legendSettings'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [{ Car:'Mustang', Brand:'Ford', count:232 },
                    { Car:'EcoSport', Brand:'Ford', count:121 },
                    { Car:'Swift', Brand:'Maruti', count:143 },
                    { Car:'Baleno', Brand:'Maruti', count:454 },
                    { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
                    { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
                    { Car:'RS7 Sportback', Brand:'Audi', count:523 }
    ],
   weightValuePath: 'count',
    equalColorValuePath:'Brand',
    legendSettings: {
        visible: true,
        mode:'Interactive',
        position:'Top',
        border:{color:'black',width:2}
    },
    leafItemSettings: {
        labelPath: 'Car',
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        colorMapping:[
            {
                value:'Ford',
                color:'green'
            },
            {
                value:'Audi',
                color:'red'
            },
            {
                value:'Maruti',
                color:'orange'
            }
        ]
     }
    }
  },
   provide:{
    treemap:[TreeMapLegend]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs37" %}

Legend size

Customize the legend size by modifying the height and width properties in the legendSettings. It accepts values in both percentage and pixel.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:equalColorValuePath='equalColorValuePath' :legendSettings='legendSettings'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [{ Car:'Mustang', Brand:'Ford', count:232 },
                    { Car:'EcoSport', Brand:'Ford', count:121 },
                    { Car:'Swift', Brand:'Maruti', count:143 },
                    { Car:'Baleno', Brand:'Maruti', count:454 },
                    { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
                    { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
                    { car:'RS7 Sportback', Brand:'Audi', count:523 }
    ],
   weightValuePath: 'count',
    equalColorValuePath:'Brand',
```

```
    legendSettings: {
        visible: true,
        height:'50px',
        width:'200px',
        position:'Top',
        border:{color:'black',width:2}
    },
    leafItemSettings: {
        labelPath: 'Car',
        colorMapping:[
            {
               value:'Ford',
               color:'green'
           },
           {
               value:'Audi',
               color:'red'
           },
           {
               value:'Maruti',
               color:'orange'
           }
        ]
     }
    }
  },
   provide:{
    treemap:[TreeMapLegend]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs38" %}

Paging support

TreeMap support legend paging, if the legend items cannot be placed within the provided height and width of the legend.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:equalColorValuePath='equalColorValuePath' :legendSettings='legendSettings'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [{ Car:'Mustang', Brand:'Ford', count:232 },
```

```
                        { Car:'EcoSport', Brand:'Ford', count:121 },
                        { Car:'Swift', Brand:'Maruti', count:143 },
                        { Car:'Baleno', Brand:'Maruti', count:454 },
                        { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
                        { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
                        { car:'RS7 Sportback', Brand:'Audi', count:523 }
    ],
   weightValuePath: 'count',
    equalColorValuePath:'Brand',
    legendSettings: {
        visible: true,
        height:'50px',
        width:'100px',
        border:{color:'black',width:2}
    },
    leafItemSettings: {
        labelPath: 'Car',
        colorMapping:[
            {
               value:'Ford',
               color:'green'
           },
           {
               value:'Audi',
               color:'red'
           },
           {
               value:'Maruti',
               color:'orange'
           }
        ]
     },
    }
  },
   provide:{
    treemap:[TreeMapLegend]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs39" %}

Legend for items excluded from color mapping

Based on the mapping ranges in the data source, get the excluded ranges from the color mapping, and show the legend with the excluded range values that are bound to the specific legend.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:rangeColorValuePath='rangeColorValuePath' :legendSettings='legendSettings'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [{ fruit:'Apple', count:5000 },
                   { fruit:'Mango', count:3000 },
                   { fruit:'Orange', count:2300 },
                   { fruit:'Banana', count:500 },
                   { fruit:'Grape', count:4300 },
                   { fruit:'Papaya', count:1200 },
                   { fruit:'Melon', count:4500 }
    ],
   weightValuePath: 'count',
    rangeColorValuePath:'count',
    legendSettings: {
        visible: true
    },
    leafItemSettings: {
        labelPath: 'fruit',
        colorMapping:[
            {
               from: 500,
               to: 2500,
               color:'green'
           },
           {
               from: 3000,
               to: 4000,
               color:'red'
           },
           {
               color:'orange'
           }
        ]
     }
    }
  },
   provide:{
    treemap:[TreeMapLegend]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs40" %}

Hide desired legend items

To enable or disable the desired legend item for each color mapping, set the showLegend property to **true** in the colorMapping.

APP.VUE

```
<template>
    <div class="control_wrapper">
```

```
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:rangeColorValuePath='rangeColorValuePath' :legendSettings='legendSettings'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [{ fruit:'Apple', count:5000 },
                    { fruit:'Mango', count:3000 },
                    { fruit:'Orange', count:2300 },
                    { fruit:'Banana', count:500 },
                    { fruit:'Grape', count:4300 },
                    { fruit:'Papaya', count:1200 },
                    { fruit:'Melon', count:4500 }
    ],
   weightValuePath: 'count',
    rangeColorValuePath:'count',
    legendSettings: {
        visible: true
    },
    leafItemSettings: {
        labelPath: 'fruit',
        colorMapping:[
            {
                from: 500,
                to: 2500,
                color:'green',
                showLegend: false,
            },
            {
                from: 3000,
                to: 4000,
                color:'red'
                showLegend: true,
            },
            {
                color:'orange',
                showLegend: true,
            }
        ]
     }
    }
  },
   provide:{
    treemap:[TreeMapLegend]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs41" %}

Copyright © 2001 -2024 Syncfusion Inc.

Hide legend items based data source value

To enable or disable the legend visibility for each item through the data source, bind the appropriate data source field name to showLegendPath property in the legendSettings.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:colorValuePath='colorValuePath' :legendSettings='legendSettings'
:weightValuePath='weightValuePath' :palette='palette'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [{ fruit:'Apple', count:5000, visibility: true , color:
'red'},
                        { fruit:'Mango', count:3000, visibility: false,
color:'blue'},
                        { fruit:'Orange', count:2300, visibility: true,
color:'green' },
                        { fruit:'Banana', count:500, visibility: false,
color:'yellow' },
                        { fruit:'Grape', count:4300, visibility: true,
color:'orange' },
                        { fruit:'Papaya', count:1200, visibility: false,
color:'pink' },
                        { fruit:'Melon', count:4500, visibility: true ,
color:'violet'}
    ],
    weightValuePath: 'count',
    colorValuePath:'color',
    palette:[],
    legendSettings: {
        visible: true,
        showLegendPath:'visibility'
    },
    leafItemSettings: {
        labelPath: 'fruit',
     }
    }
  },
   provide:{
    treemap:[TreeMapLegend]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs42" %}

Copyright © 2001 -2024 Syncfusion Inc.

Bind legend item text from data source

To show the legend item text from the data source, bind the property name from data source to the valuePath property in the legendSettings.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:colorValuePath='colorValuePath' :legendSettings='legendSettings'
:weightValuePath='weightValuePath' :palette='palette'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [{ fruit:'Apple', count:5000, visibility: true , color:
'red'},
                        { fruit:'Mango', count:3000, visibility: false,
color:'blue'},
                        { fruit:'Orange', count:2300, visibility: true,
color:'green' },
                        { fruit:'Banana', count:500, visibility: false,
color:'yellow' },
                        { fruit:'Grape', count:4300, visibility: true,
color:'orange' },
                        { fruit:'Papaya', count:1200, visibility: false,
color:'pink' },
                        { fruit:'Melon', count:4500, visibility: true ,
color:'violet'}
    ],
    weightValuePath: 'count',
    colorValuePath:'color',
    palette:[],
    legendSettings: {
        visible: true,
        valuePath:'fruit'
    },
    leafItemSettings: {
        labelPath: 'fruit',
     }
    }
  },
   provide:{
    treemap:[TreeMapLegend]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs43" %}

Copyright © 2001 -2024 Syncfusion Inc.

Hide duplicate legend items

To enable or disable the duplicate legend items, set the removeDuplicateLegend property to **true** in the legendSettings.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:colorValuePath='colorValuePath' :legendSettings='legendSettings'
:weightValuePath='weightValuePath' :palette='palette'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [{ fruit:'Apple', count:5000, visibility: true , color:
'red'},
                        { fruit:'Mango', count:3000, visibility: false,
color:'blue'},
                        { fruit:'Orange', count:2300, visibility: true,
color:'green' },
                        { fruit:'Banana', count:500, visibility: false,
color:'yellow' },
                        { fruit:'Grape', count:4300, visibility: true,
color:'orange' },
                        { fruit:'Mango', count:1200, visibility: false,
color:'pink' },
                        { fruit:'Banana', count:4500, visibility: true ,
color:'violet'}
    ],
    weightValuePath: 'count',
    colorValuePath:'color',
    palette:[],
    legendSettings: {
        visible: true,
        valuePath:'fruit',
        removeDuplicateLegend: true
    },
    leafItemSettings: {
        labelPath: 'fruit',
     }
    }
  },
   provide:{
    treemap:[TreeMapLegend]
    },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs44" %}

Copyright © 2001 -2024 Syncfusion Inc.

Legend Responsiveness

Use a responsive legend that switches positions between the right and the bottom based on the available height and width. To enable the responsive legend, set the position property to **Auto** in the legendSettings and the legend position is changed based on the available height and width.

APP.VUE

```
<template>
  <div class="control_wrapper">
    <ejs-treemap id="treemap" :dataSource='dataSource'
:weightValuePath='weightValuePath' :legendSettings='legendSettings'
      :palette='palette' :leafItemSettings='leafItemSettings'></ejs-treemap>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function () {
    return {
      dataSource: [{ fruit: 'Apple', count: 5000 },
      { fruit: 'Grape', count: 3000 },
      { fruit: 'Apple', count: 2300 },
      { fruit: 'Banana', count: 500 },
      { fruit: 'Grape', count: 4300 },
      { fruit: 'Papaya', count: 1200 }
      ],
      width: '700px',
      height: '500px',
      weightValuePath: 'count',
      palette: ['#71B081', '#5A9A77', '#498770', '#39776C', '#266665',
'#124F5E'],
      legendSettings: {
        visible: true,
        position: 'Auto'
      },
      leafItemSettings: {
        labelPath: 'fruit',
      }
    }
  },
  provide: {
    treemap: [TreeMapLegend]
  },
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs45" %}

Drilldown in Vue Treemap component

The TreeMap supports drill-down to expose the hierarchy, achieved by clicking a node. If an item is clicked in the TreeMap, it will be moved to the next level or sub level hierarchy and returned back to the previous level by clicking the node.

Copyright © 2001 -2024 Syncfusion Inc.

Perform drill-down action

The TreeMap items can be drilled by setting the enableDrillDown property to **true**.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:palette='palette' :enableDrillDown= 'enableDrillDown'
:weightValuePath='weightValuePath' :levels='levels'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
        { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }
    ],
```

Copyright © 2001 -2024 Syncfusion Inc.

```
     palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
"#009688"],
    weightValuePath: 'EmployeesCount',
    enableDrillDown: true,
    levels: [
        { groupPath: 'Country', border: { color: 'black', width: 0.5 } },
        { groupPath: 'JobDescription', border: { color: 'black', width: 0.5
} },
        { groupPath: 'JobGroup', border: { color: 'black', width: 0.5 } },
    ]
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs12" %}

On-demand data loading

All the child items are rendered during the normal drill-down process, and visible at the initial rendering of the TreeMap. But on-demand data loading, it will not render child items at initial rendering, and child nodes will be rendered during the drill-down process by setting the drillDownView property to **true**.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:palette='palette' :enableDrillDown= 'enableDrillDown'
:weightValuePath='weightValuePath' :levels='levels' :drillDownView=
'drillDownView'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
        { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }
    ],
     palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
"#009688"],
    weightValuePath: 'EmployeesCount',
    enableDrillDown: true,
    drillDownView: true,
    levels: [
        { groupPath: 'Country', border: { color: 'black', width: 0.5 } },
        { groupPath: 'JobDescription', border: { color: 'black', width: 0.5
} },
        { groupPath: 'JobGroup', border: { color: 'black', width: 0.5 } },
    ]
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs13" %}

Breadcrumb support

TreeMap items are drilled, up to any level of parent using breadcrumb navigation and the level from root parent to current level is displayed at the top of item layout. It can be enabled by using the enableBreadcrumb property to **true** and customize the breadcrumb connector using the breadcrumbConnector property. By default, -(hyphen) is the connector.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:palette='palette' :enableDrillDown= 'enableDrillDown'
:weightValuePath='weightValuePath' :levels='levels' :breadcrumbConnector=
'breadcrumbConnector' :enableBreadcrumb='enableBreadcrumb' ></ejs-treemap>
    </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
        { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }
    ],
     palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
"#009688"],
    weightValuePath: 'EmployeesCount',
    enableDrillDown: true,
    enableBreadcrumb:true,
    breadcrumbConnector: ' -> ',
    levels: [
        { groupPath: 'Country', border: { color: 'black', width: 0.5 } },
        { groupPath: 'JobDescription', border: { color: 'black', width: 0.5
} },
        { groupPath: 'JobGroup', border: { color: 'black', width: 0.5 } },
    ]
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
   }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs14" %}

Tooltip in Vue Treemap component

Tooltip is used to display details about the items in the TreeMap. When space constraints prevent us from displaying the information using Data Labels, the tooltip comes in handy.

Default tooltip

The tooltip is not visible by default, to make it visible, set the visible property in the tooltipSettings to **true** and injecting the TreeMapTooltip.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:tooltipSettings='tooltipSettings' :weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapTooltip } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
   weightValuePath: 'count',
     tooltipSettings: {
            visible: true
    },
    leafItemSettings: {
        labelPath: 'fruit'
     }
    }
  },
  provide:{
    treemap:[TreeMapTooltip]
},
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs57" %}

Copyright © 2001 -2024 Syncfusion Inc.

Format tooltip

The tooltip content is displayed by default based on the weightValuePath. In addition, to show more information in the tooltip, use the format property and define field from the data source as ${datafield}.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:tooltipSettings='tooltipSettings' :weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapTooltip } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
   weightValuePath: 'count',
     tooltipSettings: {
            visible: true,
            format:'Name:${fruit} - TotalCount:${count}'
    },
    leafItemSettings: {
        labelPath: 'fruit'
     }
    }
  },
  provide:{
    treemap:[TreeMapTooltip]
},
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs58" %}

Tooltip template

Tooltip can be rendered as a custom component using the template property in the tooltipSettings which accepts one or more UI elements as an input, that can be rendered as a part of the tooltip rendering. You can use ${datafield} as placeholder in HTML element to display the values from data source.

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:tooltipSettings='tooltipSettings' :weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import Template1 from "./LabelTemplate.vue";
import { TreeMapPlugin, TreeMapTooltip } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
let contentVue = Vue.component("contentTemplate", {
  template: '<div><p>Name : {{data.fruit}}</p><p>TotalCount :
{{data.count}}</p></div>',
  data() {
    return {
      data: {}
    };
  }
});
let contentTemplate = function() {
  return { template: contentVue };
};
export default {
  data: function() {
    return {
     dataSource: [
        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
   weightValuePath: 'count',
     tooltipSettings: {
            visible: true,
            template:contentTemplate
    },
    leafItemSettings: {
        labelPath: 'fruit'
     }
    }
  },
  provide:{
    treemap:[TreeMapTooltip]
},
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs59" %}

Copyright © 2001 -2024 Syncfusion Inc.

Selection and highlight in Vue Treemap component

Selection

Selection is used to select a particular group or item to differentiate from other items. Each item or each group can be selected and deselected while interacting with the items. The corresponding Treemap items are also selected while tapping a specific legend item, and vice versa.

The fill property is used to change the selected item color. The color and the width properties are used to customize the selected item border, and the selection is enabled by using the enable property to **true** in the selectionSettings.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap" :dataSource='dataSource' :levels='levels'
:selectionSettings='selectionSettings' :weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapSelection } from "@syncfusion/ej2-vue-
treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
        { dataType: "Import", type: "Animal products",   product: "2010",
sales: 20839332874 },
    { dataType: "Import", type: "Animal products",   product: "2011", sales:
23098635589 },
    { dataType: "Import", type: "Chemical products", product: "2010", sales:
141637951510 },
    { dataType: "Import", type: "Chemical products", product: "2011", sales:
161550338209 },
    { dataType: "Import", type: "Base metals",       product: "2010", sales:
86079439944 },
    { dataType: "Import", type: "Base metals",       product: "2011", sales:
103821671535 },
    { dataType: "Import", type: "Textile articles",       product: "2010",
sales: 97126140830 },
    { dataType: "Import", type: "Textile articles",       product: "2011",
sales: 104980750811 },
    { dataType: "Export", type: "Animal products",   product: "2010", sales:
15845503378 },
    { dataType: "Export", type: "Animal products",   product: "2011", sales:
20650111620 },
    { dataType: "Export", type: "Chemical products", product: "2010", sales:
136100054087 },
    { dataType: "Export", type: "Chemical products", product: "2011", sales:
146341672411 },
    { dataType: "Export", type: "Base metals",       product: "2010", sales:
59060592813 },
    { dataType: "Export", type: "Base metals",       product: "2011", sales:
71785882641 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    { dataType: "Export", type: "Textile articles",       product: "2010",
sales: 20982380561 },
    { dataType: "Export", type: "Textile articles",       product: "2011",
sales: 26016143783 }
    ],
    weightValuePath: 'sales',
        levels: [
            { groupPath: 'dataType', fill: '#c5e2f7', headerStyle: { size:
'16px' }, headerAlignment: 'Center', groupGap: 5 },
            { groupPath: 'product', fill: '#a4d1f2', headerAlignment:
'Center' , groupGap: 2 }
        ],
        leafItemSettings: {
            labelPath: 'type',
            fill: '#8ebfe2',
            labelPosition: 'Center'
        },
        selectionSettings: {
            enable: true,
            fill: '#58a0d3',
            border: { width: 0.3, color: 'black' },
            opacity: '1'
        },
    }
  },
  provide:{
    treemap:[ TreeMapSelection]
},
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs55" %}

Highlight

Highlight is used to highlight an item or group from other items or groups. Each item or each group can be highlighted by hovering the mouse over the items. The corresponding Treemap items are also be highlighted while hovering over a specific legend item, and vice versa.

The fill property is used to change the highlighted item color. The color and the width properties are used to customize the highlighted item border, and the highlight is enabled by setting the enable property to **true** in the highlightSettings.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap" :dataSource='dataSource' :levels='levels'
:highlightSettings='highlightSettings' :weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapHighlight } from "@syncfusion/ej2-vue-
treemap";
Vue.use(TreeMapPlugin);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
export default {
  data: function() {
    return {
     dataSource: [
        { dataType: "Import", type: "Animal products",   product: "2010",
sales: 20839332874 },
    { dataType: "Import", type: "Animal products",   product: "2011", sales:
23098635589 },
    { dataType: "Import", type: "Chemical products", product: "2010", sales:
141637951510 },
    { dataType: "Import", type: "Chemical products", product: "2011", sales:
161550338209 },
    { dataType: "Import", type: "Base metals",       product: "2010", sales:
86079439944 },
    { dataType: "Import", type: "Base metals",       product: "2011", sales:
103821671535 },
    { dataType: "Import", type: "Textile articles",       product: "2010",
sales: 97126140830 },
    { dataType: "Import", type: "Textile articles",       product: "2011",
sales: 104980750811 },
    { dataType: "Export", type: "Animal products",   product: "2010", sales:
15845503378 },
    { dataType: "Export", type: "Animal products",   product: "2011", sales:
20650111620 },
    { dataType: "Export", type: "Chemical products", product: "2010", sales:
136100054087 },
    { dataType: "Export", type: "Chemical products", product: "2011", sales:
146341672411 },
    { dataType: "Export", type: "Base metals",       product: "2010", sales:
59060592813 },
    { dataType: "Export", type: "Base metals",       product: "2011", sales:
71785882641 },
    { dataType: "Export", type: "Textile articles",       product: "2010",
sales: 20982380561 },
    { dataType: "Export", type: "Textile articles",       product: "2011",
sales: 26016143783 }
    ],
   weightValuePath: 'sales',
        levels: [
            { groupPath: 'dataType', fill: '#c5e2f7', headerStyle: { size:
'16px' }, headerAlignment: 'Center', groupGap: 5 },
            { groupPath: 'product', fill: '#a4d1f2', headerAlignment:
'Center' , groupGap: 2 }
        ],
        leafItemSettings: {
            labelPath: 'type',
            fill: '#8ebfe2',
            labelPosition: 'Center'
        },
         highlightSettings: {
            enable: true,
            fill: '#71b0dd',
            border: { width: 0.3, color: 'black' },
            opacity: '1'
        }
    }
  },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  provide:{
    treemap:[TreeMapHighlight]
},
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs56" %}

Print and export in Vue Treemap component

Print

To use the print functionality, we should inject the Print module using the provide section and set the allowPrint property to **true**. The rendered treemap can be printed directly from the browser by calling the method print.

APP.VUE

```
<template>
  <div class="control_wrapper">
    <button id="print" @click="clickPrint">Print</button>
    <ejs-treemap ref="treemap" :allowPrint='allowPrint'
:dataSource='dataSource' :weightValuePath='weightValuePath'
      :leafItemSettings='leafItemSettings'></ejs-treemap>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, Print } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function () {
    return {
      dataSource: [
        { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1 },
        { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
        { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
        { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
        { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5 },
        { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
        { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
        { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
        { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
        { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
      ],
      weightValuePath: 'GDP',
      leafItemSettings: {
        labelPath: 'State',
        labelFormat: '${State}<br>$${GDP} Trillion<br>(${percentage} %)',
        labelStyle: {
          color: '#000000'
        },
        border: {
          color: '#000000',
          width: 0.5
        }
      },
```

```
      allowPrint: true
    }
  },
  provide: {
    treemap: [Print]
  },
  methods: {
    clickPrint: function () {
      this.$refs.treemap.print();
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs51" %}

Export

Image Export

To use the image export functionality, we should inject the ImageExport module using the provide section and set the allowImageExport property to **true**. The rendered treemap can be exported as an image using the export method. The method requires two parameters: image type and file name. The treemap can be exported as an image in the following formats.

- JPEG
- PNG
- SVG

APP.VUE

```
<template>
  <div class="control_wrapper">
    <button id="export" @click="clickExport">Export</button>
    <ejs-treemap id="treemap" ref='treemap'
:allowImageExport='allowImageExport' :dataSource='dataSource'
      :weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, ImageExport } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function () {
    return {
      dataSource: [
        { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1 },
        { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
        { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
        { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
        { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5 },
        { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
        { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
        { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
        { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
      ],
      weightValuePath: 'GDP',
      leafItemSettings: {
        labelPath: 'State',
        labelFormat: '${State}<br>$${GDP} Trillion<br>(${percentage} %)',
        labelStyle: {
          color: '#000000'
        },
        border: {
          color: '#000000',
          width: 0.5
        },
      },
      allowImageExport: true
    }
  },
  provide: {
    treemap: [ImageExport]
  },
  methods: {
    clickExport: function () {
      this.$refs.treemap.export('PNG', 'TreeMap');
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs52" %}

We can get the image file as base64 string for the JPEG and PNG formats. The treemap can be exported to image as a base64 string using the export method. There are four parameters required: image type, file name, orientation of the exported PDF document which must be set as **null** for image export and finally **allowDownload** which should be set as **false** to return base64 string.

APP.VUE

```
<template>
  <div class="control_wrapper">
    <button id="export" @click="clickExport">Export</button>
    <ejs-treemap id="treemap" ref='treemap'
:allowImageExport='allowImageExport' :dataSource='dataSource'
      :weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, ImageExport } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function () {
    return {
      dataSource: [
        { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1 },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
        { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
        { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
        { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5 },
        { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
        { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
        { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
        { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
        { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
      ],
      weightValuePath: 'GDP',
      leafItemSettings: {
        labelPath: 'State',
        labelFormat: '${State}<br>$${GDP} Trillion<br>(${percentage} %)',
        labelStyle: {
          color: '#000000'
        },
        border: {
          color: '#000000',
          width: 0.5
        }
      },
      allowImageExport: true
    }
  },
  provide: {
    treemap: [ImageExport]
  },
  methods: {
    clickExport: function () {
      this.$refs.treemap.export('PNG', 'TreeMap', null, false
      ).then((data) => {
        var base64 = data;
        document.writeln(base64);
      });
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs53" %}

PDF Export

To use the PDF export functionality, we should inject the PdfExport module using the provide section and set the allowPdfExport property to **true**. The rendered treemap can be exported as PDF using the export method. The export method requires three parameters: file type, file name and orientation of the PDF document. The orientation setting is optional and **0** indicates portrait and **1** indicates landscape.

APP.VUE

```
<template>
  <div class="control_wrapper">
    <button id="export" @click="clickExport">export</button>
    <ejs-treemap id="treemap" ref='treemap' :allowPdfExport='allowPdfExport'
:dataSource='dataSource'
```

```
      :weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, PdfExport } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function () {
    return {
      dataSource: [
        { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1 },
        { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
        { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
        { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
        { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5 },
        { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
        { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
        { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
        { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
        { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
      ],
      weightValuePath: 'GDP',
      leafItemSettings: {
        labelPath: 'State',
        labelFormat: '${State}<br>$${GDP} Trillion<br>(${percentage} %)',
        labelStyle: {
          color: '#000000'
        },
        border: {
          color: '#000000',
          width: 0.5
        }
      },
      allowPdfExport: true
    }
  },
  provide: {
    treemap: [PdfExport]
  },
  methods: {
    clickExport: function (args) {
      this.$refs.treemap.export('PDF', 'TreeMap', 0);
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs54" %}

The exporting of the treemap as base64 string is not supported in the PDF export.

Accessibility in Vue Treemap component

TreeMap has built-in accessibility features like screen reading and WAI-ARIA attributes.

Copyright © 2001 -2024 Syncfusion Inc.

WAI-ARIA attributes

The TreeMap component follows the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the TreeMap component:

| Attributes | Purpose |

| --- | --- |

| role=region | It specifies the TreeMap areas that do not support interactive functions like selection and highlight. |

| role=button | It specifies the TreeMap areas where interactive functions such as selection and highlight are available. |

| aria-label | Provides an accessible name for the data labels, legend title, and legend item labels. |

Screen reading in TreeMap

Accessibility in the TreeMap component ensures that all users, regardless of ability or disability, can use screen reading. The following TreeMap elements will be read aloud using screen reading software, such as Narrator for Windows.

| Elements | Description |

| --- | --- |

| Data labels | Reads the labels displayed on leaf items of the TreeMap. |

| Legend title | Reads the title of the legend in the TreeMap. |

| Legend item label | Reads the label of the legend item in the TreeMap. |

Ensuring accessibility

The TreeMap component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the TreeMap component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the TreeMap component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/tree-map.html" %}

See also

- Accessibility in Syncfusion Vue components

Internationalization in Vue Treemap component

The TreeMap control supports internationalization for the following elements:

- Data label
- Tooltip

For more information about number and date formatter, refer to internationalization.

<!-- markdownlint-disable MD036 -->

Copyright © 2001 -2024 Syncfusion Inc.

Globalization

Globalization is the process of designing and developing a component that works in different cultures/locales. Internationalization library is used to globalize number, date, and time values in the TreeMap control using the format property in the TreeMap.

Numeric format

In the following code example, tooltip is globalized to Deutsch culture.

```
<template>
<div class="control_wrapper">
<ejs-treemap id="treemap" :dataSource='dataSource' :weightValuePath='weightValuePath' :format='format' :leafItemSettings='leafItemSettings' useGroupingSeparator='true'></ejs-treemap>
</div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
data: function() {
return {
format:'c',
dataSource: [
{State:"United States", GDP:17946, percentage:11.08, Rank:1},
{State:"China", GDP:10866, percentage: 28.42, Rank:2},
{State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
{State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
{State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
{State:"France", GDP:2421, percentage:-9.69, Rank:6},
{State:"India", GDP:2073, percentage:13.65, Rank:7},
{State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
{State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
{State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
weightValuePath: 'GDP',
```

Copyright © 2001 -2024 Syncfusion Inc.

leafItemSettings: {

labelPath: 'GDP',

}

}

}

}

</script>

`

Right-to-left rendering

The TreeMap control supports right-to-left rendering for all its elements such as nodes, tooltip, data labels, and legends.

Legend with Rtl support

If the enableRtl property is set to **true**, then the legend icon will be rendered on the right and the legend text will be rendered on the left of the legend icon.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:legendSettings='legendSettings' :colorValuePath ='colorValuePath'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
                        { Car:'Mustang', Brand:'Ford', count:232, color:
'#71B081' },
                        { Car:'EcoSport', Brand:'Ford', count:121,  color:
'#5A9A77' },
                        { Car:'Swift', Brand:'Maruti', count:143, color:
'#498770' },
                        { Car:'Baleno', Brand:'Maruti', count:454, color:
'#39776C' },
                        { Car:'Vitara Brezza', Brand:'Maruti', count:545 ,
color: '#266665' },
                        { Car:'A3 Cabriolet', Brand:'Audi',count:123, color:
'#124F5E' }
    ],
  weightValuePath: 'count',
    colorValuePath: 'color',
    enableRtl:true,
    legendSettings: {
```

```
        visible: true,
        position:'Top'
    },
    leafItemSettings: {
        labelPath: 'Car'
     },
    }
  },
  provide:{
    treemap:[TreeMapLegend]
},
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs20" %}

Tooltip with Rtl support

If the enableRtl property is set to **true**, the tooltip data will be rendered in reverse direction.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:palette='palette' :tooltipSettings='tooltipSettings'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapTooltip } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
                        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
  weightValuePath: 'count',
    palette:['#71B081','#5A9A77', '#498770', '#39776C',
'#266665','#124F5E'],
     tooltipSettings: {
            visible: true,
            format:'${count} : ${fruit}'
    },
    leafItemSettings: {
        labelPath: 'fruit'
     }
    }
```

```
  },
  provide:{
    treemap:[ TreeMapTooltip]
},
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs21" %}

Treemap Item Rendering Direction

The direction of TreeMap item is `TopLeftBottomRight` by default and customize the rendering direction of the TreeMap item by setting the renderDirection property.

The TreeMap can be rendered in the following directions:

- TopLeftBottomRight
- TopRightBottomLeft
- BottomRightTopLeft
- BottomLeftTopRight

The following example demonstrate, how to render the treemap in the RTL direction with `TopLeftBottomRight`.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:palette='palette' :renderDirection='renderDirection'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
                        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
    weightValuePath: 'count',
    palette:['#71B081','#5A9A77', '#498770', '#39776C',
'#266665','#124F5E'],
    renderDirection:'TopLeftBottomRight',
    leafItemSettings: {
        labelPath: 'fruit'
```

```
      }
    }
  },
  provide:{
    treemap:[TreeMapLegend]
},
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs22" %}

The following example demonstrate, how to render the treemap in the RTL direction with TopRightBottomLeft.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:palette='palette' :renderDirection='renderDirection'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
                        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
 weightValuePath: 'count',
    palette:['#71B081','#5A9A77', '#498770', '#39776C',
'#266665','#124F5E'],
    renderDirection:'TopRightBottomLeft',
    leafItemSettings: {
        labelPath: 'fruit'
     }
    }
  },
  provide:{
    treemap:[TreeMapLegend]
},
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs23" %}

The following example demonstrate, how to render the treemap in the RTL direction with BottomRightTopLeft.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:palette='palette' :renderDirection='renderDirection'
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
                        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
    weightValuePath: 'count',
    palette:['#71B081','#5A9A77', '#498770', '#39776C',
'#266665','#124F5E'],
    renderDirection:'BottomRightTopLeft',
    leafItemSettings: {
        labelPath: 'fruit'
     }
    }
  },
  provide:{
    treemap:[TreeMapLegend]
},
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs24" %}

The following example demonstrate, how to render the treemap in the RTL direction with BottomLeftTopRight.

APP.VUE

```
<template>
    <div class="control_wrapper">
        <ejs-treemap id="treemap"  :dataSource='dataSource'
:palette='palette' :renderDirection='renderDirection'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
:weightValuePath='weightValuePath'
:leafItemSettings='leafItemSettings'></ejs-treemap>
    </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin, TreeMapLegend } from "@syncfusion/ej2-vue-treemap";
Vue.use(TreeMapPlugin);
export default {
  data: function() {
    return {
     dataSource: [
                        { fruit:'Apple', count:5000 },
                        { fruit:'Mango', count:3000 },
                        { fruit:'Orange', count:2300 },
                        { fruit:'Banana', count:500 },
                        { fruit:'Grape', count:4300 },
                        { fruit:'Papaya', count:1200 },
                        { fruit:'Melon', count:4500 }
    ],
    weightValuePath: 'count',
    palette:['#71B081','#5A9A77', '#498770', '#39776C',
'#266665','#124F5E'],
    renderDirection:'BottomLeftTopRight',
    leafItemSettings: {
        labelPath: 'fruit'
     }
    }
  },
  provide:{
    treemap:[TreeMapLegend]
},
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs25" %}

How To

Drilldown in Vue Treemap component

<!-- markdownlint-disable MD033 -->

Customize the header for treemap drilldown

Yon can add a header element as and customize it to show the population of a particular country or continent on treemap drill-down.

To customize the header for treemap drill-down, follow the given steps:

Step 1:

<!-- markdownlint-disable MD031 -->

Initialize the treemap and enable the drill-down option.

`

<template>

Copyright © 2001 -2024 Syncfusion Inc.

```
<div>
<div id="header" style="background-color: #179bd7"></div>
<div class='control_wrapper'>
<ejs-treemap id='container' :dataSource="dataSource" weightValuePath="Population" format="n"
useGroupingSeparator="true" enableDrillDown="true" :palette="palette"
:leafItemSettings='leafItemSettings' :levels='levels'>
</ejs-treemap>
</div>
</div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from '@syncfusion/ej2-vue-treemap';
import { data } from './datasource.js';
Vue.use(TreeMapPlugin);
export default {
data () {
return {
dataSource:  data,
palette: ["#9999ff", "#CCFF99", "#FFFF99", "#FF9999", "#FF99FF", "#FFCC66"],
levels: [
{
groupPath: 'Continent', border: { color: 'black', width: 0.5 },
},
{
groupPath: 'States', border: { color: 'black', width: 0.5 },
},
{
groupPath: 'Region', border: { color: 'black', width: 0.5 },
}
],
leafItemSettings: {
labelPath:'Name',
```

Copyright © 2001 -2024 Syncfusion Inc.

showLabels: false

}

}

},

}

</script>

`

Step 2:

Show the population of a particular continent in the treemap loaded event. In this event, you can get the header element.

`javascript

loaded:function(args) {

let header = document.getElementById('header');

var population = 0;

for (let i = 0; i < args.treemap.layout.renderItems[0]['parent'].Continent.length; i++) {

population += +(args.treemap.layout.renderItems[0]['parent'].Continent[i]['data'].Population);

}

header.innerHTML = 'Continent - Population : ' + population

}

`

Step 3:

Customize the population for drilled countries or states in the header element when drill-down the treemap. The drillEnd event will be triggered when treemap is drilled.

APP.VUE

```
<template>
<div>
    <div id="header" style="background-color: #179bd7"></div>
      <div class='control_wrapper'>
         <ejs-treemap id='container' :dataSource="dataSource"
weightValuePath="Population" format="n"
                    useGroupingSeparator="true" enableDrillDown="true"
:palette="palette"
         :leafItemSettings='leafItemSettings' :levels='levels'
:drillEnd='drillEnd' :loaded="loaded">
         </ejs-treemap>
      </div>
  </div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { TreeMapPlugin } from '@syncfusion/ej2-vue-treemap';
import { data } from './datasource.js';
Vue.use(TreeMapPlugin);
export default {
   data () {
    return {
      dataSource:  data,
        palette: ["#9999ff", "#CCFF99", "#FFFF99", "#FF9999", "#FF99FF",
"#FFCC66"],
        levels: [
            {
                groupPath: 'Continent', border: { color: 'black', width: 0.5
},
            },
            {
              groupPath: 'States', border: { color: 'black', width: 0.5 },
            },
            {
              groupPath: 'Region', border: { color: 'black', width: 0.5 },
            }
        ],
        leafItemSettings: {
          labelPath:'Name',
          showLabels: false
        }
    }
  },
  methods:{
    drillEnd:function(args){
        let header = document.getElementById('header');
        let layout =
document.getElementById("container_TreeMap_Squarified_Layout");
        let population = 0;
        if (args.treemap.layout.renderItems[0]['isDrilled']) {
            for (let i = 0; i < args.treemap.layout.renderItems.length; i++)
{
                population +=
+(args.treemap.layout.renderItems[i]['data'].Population);
            }
            header.innerHTML =
layout.children[0].children[1].innerHTML.split(']')[1] + ' - ' + population;
        }
        else if (args.treemap.layout.renderItems[0]['parent'].Continent) {
            for (let i = 0; i <
args.treemap.layout.renderItems[0]['parent'].Continent.length; i++) {
                population +=
+(args.treemap.layout.renderItems[0]['parent'].Continent[i]['data'].Populati
on);
            }
            header.innerHTML = 'Continent - Population : ' + population;
        } else {
            population =
args.treemap.layout.renderItems[0]['data'].Population;
            header.innerHTML =
layout.children[0].children[1].innerHTML.split(']')[1] + ' - Population : '
+ population;
        }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    },
    loaded:function(args) {
      let header = document.getElementById('header');
        var population = 0;
        for (let i = 0; i <
args.treemap.layout.renderItems[0]['parent'].Continent.length; i++) {
            population +=
+(args.treemap.layout.renderItems[0]['parent'].Continent[i]['data'].Populati
on);
        }
        header.innerHTML = 'Continent - Population : ' + population
    }
}
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs19" %}

Drilldown with label in Vue Treemap component

Add label template with drill down

Yon can add a label template as element to the treemap control when using the label template. To add a label template to the treemap control, you have to hide another labels by setting the showLabels property to **false** in leafItemSettings to show only the label template.

To add label template to treemap drilldown, follow the given steps:

Step 1:

Create a treemap control and enable the drill-down option.

`

<template>

<ejs-treemap id='container' :dataSource="dataSource" weightValuePath="Sales"
enableDrillDown="true" :palette="palette"

:leafItemSettings='leafItemSettings' :levels='levels' >

</ejs-treemap>

</template>

<script>

import Vue from 'vue';

import { TreeMapPlugin } from '@syncfusion/ej2-vue-treemap';

Vue.use(TreeMapPlugin);

export default {

data () {

return {

Copyright © 2001 -2024 Syncfusion Inc.

```
dataSource:  [
{ Continent: "China", Company: "Volkswagen", Sales: 3005994 },
{ Continent: "China", Company: "General Motors", Sales: 1230044 },
{ Continent: "China", Company: "Honda", Sales: 1197023 },
{ Continent: "United States", Company: "General Motors", Sales: 3042775 },
{ Continent: "United States", Company: "Ford", Sales: 2599193 },
{ Continent: "United States", Company: "Toyota", Sales: 2449587 },
{ Continent: "Japan", Company: "Toyota", Sales: 1527977 },
{ Continent: "Japan", Company: "Honda", Sales: 706982 },
{ Continent: "Japan", Company: "Suzuki", Sales: 623041 },
{ Continent: "Germany", Company: "Volkswagen", Sales: 655977 },
{ Continent: "Germany", Company: "Mercedes", Sales: 310845 },
{ Continent: "Germany", Company: "BMW", Sales: 261931 },
{ Continent: "United Kingdom", Company: "Ford ", Sales: 319442 },
{ Continent: "United Kingdom", Company: "Vauxhall", Sales: 251146 },
{ Continent: "United Kingdom", Company: "Volkswagen", Sales: 206994 }
],
palette: ['white'],
levels: [
{
groupPath: 'Continent', border: { color: 'black', width: 0.5 },
},
{
groupPath: 'Company', border: { color: 'black', width: 0.5 },
}
],
leafItemSettings: {
showLabels: false,
labelTemplate: '<div style="background-color: red">${Company}</div>',
templatePosition: 'Center'
}
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

}

</script>

`

Step 2:

Add the label template in the leafItemSettings options, and then set the showLabels property to **false** to hide another labels and show only label template.

APP.VUE

```
<template>
      <div class='control_wrapper'>
         <ejs-treemap id='container' :dataSource="dataSource"
weightValuePath="Sales" enableDrillDown="true" :palette="palette"
         :leafItemSettings='leafItemSettings' :levels='levels'
:drillStart='drillStart' >
         </ejs-treemap>
      </div>
</template>
<script>
import Vue from 'vue';
import { TreeMapPlugin } from '@syncfusion/ej2-vue-treemap';
Vue.use(TreeMapPlugin);
export default {
   data () {
    return {
      dataSource:  [
            { Continent: "China", Company: "Volkswagen", Sales: 3005994 },
            { Continent: "China", Company: "General Motors", Sales: 1230044
},
            { Continent: "China", Company: "Honda", Sales: 1197023 },
            { Continent: "United States", Company: "General Motors", Sales:
3042775 },
            { Continent: "United States", Company: "Ford", Sales: 2599193 },
            { Continent: "United States", Company: "Toyota", Sales: 2449587
},
            { Continent: "Japan", Company: "Toyota", Sales: 1527977 },
            { Continent: "Japan", Company: "Honda", Sales: 706982 },
            { Continent: "Japan", Company: "Suzuki", Sales: 623041 },
            { Continent: "Germany", Company: "Volkswagen", Sales: 655977 },
            { Continent: "Germany", Company: "Mercedes", Sales: 310845 },
            { Continent: "Germany", Company: "BMW", Sales: 261931 },
            { Continent: "United Kingdom", Company: "Ford ", Sales: 319442
},
            { Continent: "United Kingdom", Company: "Vauxhall", Sales:
251146 },
            { Continent: "United Kingdom", Company: "Volkswagen", Sales:
206994 }
        ],
        palette: ['white'],
        levels: [
            {
                groupPath: 'Continent', border: { color: 'black', width: 0.5
},
            },
```

```
            {
              groupPath: 'Company', border: { color: 'black', width: 0.5 },
            }
        ],
        leafItemSettings: {
          showLabels: false,
          labelTemplate: '<div style="background-color:
red">${Company}</div>',
          templatePosition: 'Center'
        }
    }
  },
  methods:{
    drillStart:function(args){
        let labelElementGroup =
document.getElementById('container_Label_Template_Group');
        labelElementGroup.remove();
    }
  }
}
</script>
```

{% previewsample "page.domainurl/code-snippet/treemap/getting-started-cs18" %}

TreeView

Getting Started with the Vue TreeView Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue TreeView component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

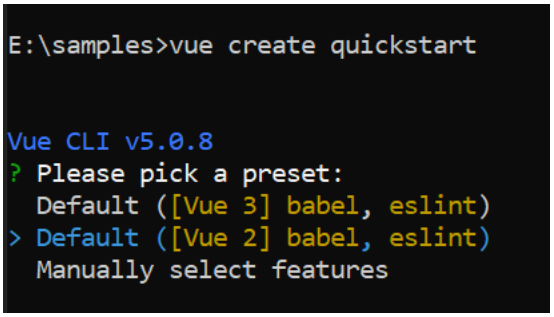
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Adding syncfusion packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue TreeView component as an example. Install the @syncfusion/ej2-vue-navigations package by running the following command:

`bash

npm install @syncfusion/ej2-vue-navigations --save

`

or

`bash

yarn add @syncfusion/ej2-vue-navigations

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the TreeView component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

Add Syncfusion Vue component

Follow the below steps to add the Vue TreeView component using Composition API or Options API:

1\. First, import and register the TreeView component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { TreeViewComponent as EjsTreeview } from "@syncfusion/ej2-vue-
navigations";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { TreeViewComponent } from "@syncfusion/ej2-vue-navigations";
export default {
components: {
'ejs-treeview': TreeViewComponent
}
}
</script>
```

Binding data source

TreeView can load data either from local data sources or remote data services. This can be done using the dataSource property that is a member of the fields property. The dataSource property supports array of JavaScript objects and DataManager. Here, an array of JSON values is passed to the TreeView component.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div id="app">
<div class="control_wrapper">
<ejs-treeview id='treeview' :fields="fields"></ejs-treeview>
</div>
</div>
</template>
<script setup>
import { TreeViewComponent } from "@syncfusion/ej2-vue-navigations";
const data =  [
{
nodeId: '01', nodeText: 'Music',
nodeChild: [
{ nodeId: '01-01', nodeText: 'Gouttes.mp3' }
]
},
{
nodeId: '02', nodeText: 'Videos', expanded: true,
nodeChild: [
{ nodeId: '02-01', nodeText: 'Naturals.mp4' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ nodeId: '02-02', nodeText: 'Wild.mpeg' },
]
},
{
nodeId: '03', nodeText: 'Documents',
nodeChild: [
{ nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
{ nodeId: '03-02', nodeText: 'Global Water, Sanitation, & Hygiene.docx' },
{ nodeId: '03-03', nodeText: 'Global Warming.ppt' },
{ nodeId: '03-04', nodeText: 'Social Network.pdf' },
{ nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
]
},
];
const fields = { dataSource: data, id: "nodeId", text: "nodeText", child:
"nodeChild" };
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div class="control_wrapper">
<ejs-treeview id='treeview' :fields='fields'></ejs-treeview>
</div>
</template>
<script>
import { TreeViewComponent } from "@syncfusion/ej2-vue-navigations";
var dataSource =  [
{
nodeId: '01', nodeText: 'Music',
nodeChild: [
{ nodeId: '01-01', nodeText: 'Gouttes.mp3' }
]
},
{
nodeId: '02', nodeText: 'Videos', expanded: true,
nodeChild: [
{ nodeId: '02-01', nodeText: 'Naturals.mp4' },
{ nodeId: '02-02', nodeText: 'Wild.mpeg' },
]
},
{
nodeId: '03', nodeText: 'Documents',
nodeChild: [
{ nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
{ nodeId: '03-02', nodeText: 'Global Water, Sanitation, & Hygiene.docx' },
{ nodeId: '03-03', nodeText: 'Global Warming.ppt' },
{ nodeId: '03-04', nodeText: 'Social Network.pdf' },
{ nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
]
},
];
export default {
data: function() {
return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
fields: { dataSource: data, id: 'nodeId', text: 'nodeText', child:
'nodeChild' },
};
}
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div id="app">
    <div class="control_wrapper">
      <ejs-treeview id='treeview' :fields="fields"></ejs-treeview>
    </div>
  </div>
</template>
<script setup>
import { TreeViewComponent as EjsTreeview } from "@syncfusion/ej2-vue-
navigations";
const data = [
  {
    nodeId: '01', nodeText: 'Music',
    nodeChild: [
      { nodeId: '01-01', nodeText: 'Gouttes.mp3' }
    ]
  },
  {
    nodeId: '02', nodeText: 'Videos', expanded: true,
    nodeChild: [
      { nodeId: '02-01', nodeText: 'Naturals.mp4' },
      { nodeId: '02-02', nodeText: 'Wild.mpeg' },
    ]
  },
  {
    nodeId: '03', nodeText: 'Documents',
    nodeChild: [
      { nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
      { nodeId: '03-02', nodeText: 'Global Water, Sanitation, &
Hygiene.docx' },
      { nodeId: '03-03', nodeText: 'Global Warming.ppt' },
      { nodeId: '03-04', nodeText: 'Social Network.pdf' },
      { nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
    ]
  },
];
const fields = { dataSource: data, id: "nodeId", text: "nodeText", child:
"nodeChild" };
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

OPTIONS API (~SRC/APP.VUE)

```
<template>
  <div id="app">
    <div class="control_wrapper">
      <ejs-treeview id='treeview' :fields="fields"></ejs-treeview>
    </div>
  </div>
</template>
<script>
import { TreeViewComponent } from "@syncfusion/ej2-vue-navigations";
var dataSource = [
  {
    nodeId: '01', nodeText: 'Music',
    nodeChild: [
      { nodeId: '01-01', nodeText: 'Gouttes.mp3' }
    ]
  },
  {
    nodeId: '02', nodeText: 'Videos', expanded: true,
    nodeChild: [
      { nodeId: '02-01', nodeText: 'Naturals.mp4' },
      { nodeId: '02-02', nodeText: 'Wild.mpeg' },
    ]
  },
  {
    nodeId: '03', nodeText: 'Documents',
    nodeChild: [
      { nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
      { nodeId: '03-02', nodeText: 'Global Water, Sanitation, &
Hygiene.docx' },
      { nodeId: '03-03', nodeText: 'Global Warming.ppt' },
      { nodeId: '03-04', nodeText: 'Social Network.pdf' },
      { nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
    ]
  }
];
export default {
  components: {
    'ejs-treeview': TreeViewComponent
  },
  data: function () {
    return {
      fields: { dataSource: dataSource, id: 'nodeId', text: 'nodeText',
child: 'nodeChild' }
    }
  }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

Run the application

To run the application, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

{% previewsample "page.domainurl/code-snippet/treeview/getting-started-cs1" %}

You can also explore our Vue TreeView example to knows how to present and manipulate data.

Getting Started with the Vue TreeView Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue TreeView component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

Copyright © 2001 -2024 Syncfusion Inc.

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue TreeView component as an example. To use the Vue TreeView component in the project, the @syncfusion/ej2-vue-navigations package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-navigations --save

`

or

`bash

yarn add @syncfusion/ej2-vue-navigations

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the TreeView component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
</style>
```

Note: TreeView component use other Syncfusion components too, the dependent component's CSS references need to be added for using all the TreeView functionalities.

Add Syncfusion Vue component

Follow the below steps to add the Vue TreeView component using Composition API or Options API:

1.First, import and register the TreeView component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

Copyright © 2001 -2024 Syncfusion Inc.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { TreeViewComponent as EjsTreeview } from "@syncfusion/ej2-vue-
navigations";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { TreeViewComponent  } from "@syncfusion/ej2-vue-navigations";
//Component registration
export default {
name: "App",
components: {
"ejs-treeview": TreeViewComponent
}
}
</script>
```

2.Add the component definition in template section.

~/SRC/APP.VUE

```
<template>
<ejs-treeview id='treeview' :fields='fields'></ejs-treeview>
</template>
```

3.Declare the values for the dataSource property in the script section.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
const fields = { dataSource: data, id: "nodeId", text: "nodeText", child:
"nodeChild" };
const data =  [
{
nodeId: '01', nodeText: 'Music',
nodeChild: [
{ nodeId: '01-01', nodeText: 'Gouttes.mp3' }
]
},
{
nodeId: '02', nodeText: 'Videos', expanded: true,
nodeChild: [
{ nodeId: '02-01', nodeText: 'Naturals.mp4' },
{ nodeId: '02-02', nodeText: 'Wild.mpeg' },
]
},
{
nodeId: '03', nodeText: 'Documents',
nodeChild: [
{ nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
{ nodeId: '03-02', nodeText: 'Global Water, Sanitation, & Hygiene.docx' },
{ nodeId: '03-03', nodeText: 'Global Warming.ppt' },
{ nodeId: '03-04', nodeText: 'Social Network.pdf' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
]
},
];
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
var data =  [
{
nodeId: '01', nodeText: 'Music',
nodeChild: [
{ nodeId: '01-01', nodeText: 'Gouttes.mp3' }
]
},
{
nodeId: '02', nodeText: 'Videos', expanded: true,
nodeChild: [
{ nodeId: '02-01', nodeText: 'Naturals.mp4' },
{ nodeId: '02-02', nodeText: 'Wild.mpeg' },
]
},
{
nodeId: '03', nodeText: 'Documents',
nodeChild: [
{ nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
{ nodeId: '03-02', nodeText: 'Global Water, Sanitation, & Hygiene.docx' },
{ nodeId: '03-03', nodeText: 'Global Warming.ppt' },
{ nodeId: '03-04', nodeText: 'Social Network.pdf' },
{ nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
]
},
];
data() {
return {
fields: { dataSource: data, id: 'nodeId', text: 'nodeText', child:
'nodeChild' },
};
}
</script>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-treeview id='treeview' :fields='fields'></ejs-treeview>
</template>
<script setup>
import { TreeViewComponent as EjsTreeview } from "@syncfusion/ej2-vue-
navigations";
const fields = { dataSource: data, id: "nodeId", text: "nodeText", child:
"nodeChild" };
const data =  [
{
```

Copyright © 2001 -2024 Syncfusion Inc.

```
nodeId: '01', nodeText: 'Music',
nodeChild: [
{ nodeId: '01-01', nodeText: 'Gouttes.mp3' }
]
},
{
nodeId: '02', nodeText: 'Videos', expanded: true,
nodeChild: [
{ nodeId: '02-01', nodeText: 'Naturals.mp4' },
{ nodeId: '02-02', nodeText: 'Wild.mpeg' },
]
},
{
nodeId: '03', nodeText: 'Documents',
nodeChild: [
{ nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
{ nodeId: '03-02', nodeText: 'Global Water, Sanitation, & Hygiene.docx' },
{ nodeId: '03-03', nodeText: 'Global Warming.ppt' },
{ nodeId: '03-04', nodeText: 'Social Network.pdf' },
{ nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
]
},
];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-treeview id='treeview' :fields='fields'></ejs-treeview>
</template>
<script>
import { TreeViewComponent  } from "@syncfusion/ej2-vue-navigations";
var data =  [
{
nodeId: '01', nodeText: 'Music',
nodeChild: [
{ nodeId: '01-01', nodeText: 'Gouttes.mp3' }
]
},
{
nodeId: '02', nodeText: 'Videos', expanded: true,
nodeChild: [
{ nodeId: '02-01', nodeText: 'Naturals.mp4' },
{ nodeId: '02-02', nodeText: 'Wild.mpeg' },
]
},
{
nodeId: '03', nodeText: 'Documents',
nodeChild: [
```

Copyright © 2001 -2024 Syncfusion Inc.

```
{ nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
{ nodeId: '03-02', nodeText: 'Global Water, Sanitation, & Hygiene.docx' },
{ nodeId: '03-03', nodeText: 'Global Warming.ppt' },
{ nodeId: '03-04', nodeText: 'Social Network.pdf' },
{ nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
]
},
];
// Component registration
export default {
name: "App",
// Declaring component and its directives
components: {
"ejs-treeview": TreeViewComponent
},
// Bound properties declarations
data() {
return {
fields: { dataSource: data, id: 'nodeId', text: 'nodeText', child:
'nodeChild' },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

or

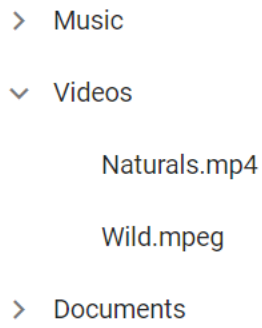
`bash

yarn run dev

`

The output will appear as follows:

Copyright © 2001 -2024 Syncfusion Inc.



CheckBox

The TreeView component allows you to check more than one node in TreeView without affecting the UI's appearance by enabling the showCheckBox property. When this property is enabled, checkbox appears before each TreeView node text.

- If one of the child nodes is not in a checked state, then the parent node will be in an intermediate state.
- If all the child nodes are in checked state, then the parent node's state will also be checked.
- If a parent node is checked, then all the child nodes' state will also be checked.

In the following example, the showCheckBox property is enabled.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<ejs-treeview id='treeview' :fields='fields' :showCheckBox='true'></ejs-
treeview>
</template>
<script setup>
import { TreeViewComponent as EjsTreeview } from "@syncfusion/ej2-vue-
navigations";
const fields = { dataSource: data, id: "nodeId", text: "nodeText", child:
"nodeChild" };
const data =  [
{
nodeId: '01', nodeText: 'Music',
nodeChild: [
{ nodeId: '01-01', nodeText: 'Gouttes.mp3' }
]
},
{
nodeId: '02', nodeText: 'Videos', expanded: true,
nodeChild: [
{ nodeId: '02-01', nodeText: 'Naturals.mp4' },
{ nodeId: '02-02', nodeText: 'Wild.mpeg' },
]
},
{
nodeId: '03', nodeText: 'Documents',
nodeChild: [
{ nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
{ nodeId: '03-02', nodeText: 'Global Water, Sanitation, & Hygiene.docx' },
```

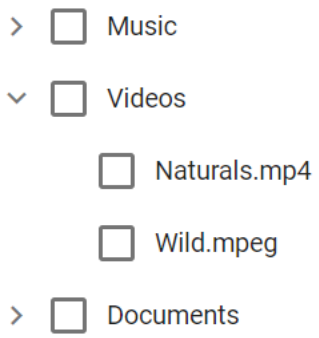
Copyright © 2001 -2024 Syncfusion Inc.

```
{ nodeId: '03-03', nodeText: 'Global Warming.ppt' },
{ nodeId: '03-04', nodeText: 'Social Network.pdf' },
{ nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
]
},
];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<ejs-treeview id='treeview' :fields='fields' :showCheckBox='true'></ejs-
treeview>
</template>
<script>
import { TreeViewComponent  } from "@syncfusion/ej2-vue-navigations";
var data =  [
{
nodeId: '01', nodeText: 'Music',
nodeChild: [
{ nodeId: '01-01', nodeText: 'Gouttes.mp3' }
]
},
{
nodeId: '02', nodeText: 'Videos', expanded: true,
nodeChild: [
{ nodeId: '02-01', nodeText: 'Naturals.mp4' },
{ nodeId: '02-02', nodeText: 'Wild.mpeg' },
]
},
{
nodeId: '03', nodeText: 'Documents',
nodeChild: [
{ nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
{ nodeId: '03-02', nodeText: 'Global Water, Sanitation, & Hygiene.docx' },
{ nodeId: '03-03', nodeText: 'Global Warming.ppt' },
{ nodeId: '03-04', nodeText: 'Social Network.pdf' },
{ nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
]
},
];
// Component registration
export default {
name: "App",
// Declaring component and its directives
components: {
"ejs-treeview": TreeViewComponent
},
// Bound properties declarations
```

```
data() {
return {
fields: { dataSource: data, id: 'nodeId', text: 'nodeText', child:
'nodeChild' },
};
}
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
</style>
```



For migrating from Vue 2 to Vue 3, refer to the migration documentation.

See also

- Getting Started with Vue UI Components using Composition API and TypeScript
- Getting Started with Vue UI Components using Options API and TypeScript

Data binding in Vue Treeview component

The TreeView component provides the option to load data either from local data sources or from remote data services. This can be done

through `dataSource` property that is a member of the fields property. The `dataSource` property supports array of JavaScript objects and **DataManager**. It also supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of `DataManager` adaptors.

TreeView has `load on demand` (Lazy load), by default. It reduces the bandwidth size when consuming huge data. It loads first level nodes initially, and when parent node is expanded, loads the child nodes based on the `parentID/child` member.

By default, the `loadOnDemand` is set to true. By disabling this property, all the tree nodes are rendered at the beginning itself.

You can use the dataBound event to perform actions. This event will be triggered once the data source is populated in the TreeView.

Copyright © 2001 -2024 Syncfusion Inc.

Local data

To bind local data to the TreeView, you can assign a JavaScript object array to the dataSource property. The TreeView component requires three fields (ID, text, and parentID) to render local data source. When mapper fields are not specified, it takes the default values as the mapping fields. Local data source can also be provided as an instance of the DataManager. It supports two kinds of local data binding methods.

- Hierarchical data
- Self-referential data

Hierarchical data

TreeView can be populated with hierarchical data source that contains nested array of JSON objects. You can directly assign hierarchical data to the dataSource property, and map all the field members with corresponding keys from the hierarchical data to fields property.

In the following example, **code**, **name**, and **countries** columns from hierarchical data have been mapped to **id**, **child**,

and **text** fields, respectively.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields"></ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
        {
            code: 'AF', name: 'Africa', countries: [
                { code: 'NGA', name: 'Nigeria' },
                { code: 'EGY', name: 'Egypt' },
                { code: 'ZAF', name: 'South Africa' }
            ]
        },
        {
            code: 'AS', name: 'Asia', countries: [
                { code: 'CHN', name: 'China' },
                { code: 'IND', name: 'India', selected: true },
                { code: 'JPN', name: 'Japan' }
            ]
        },
        {
            code: 'EU', name: 'Europe', countries: [
                { code: 'DNK', name: 'Denmark' },
                { code: 'FIN', name: 'Finland' },
                { code: 'AUT', name: 'Austria' }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            ]
        },
        {
            code: 'NA', name: 'North America', countries: [
                { code: 'USA', name: 'United States of America' },
                { code: 'CUB', name: 'Cuba' },
                { code: 'MEX', name: 'Mexico' }
            ]
        },
        {
            code: 'SA', name: 'South America', countries: [
                { code: 'BRA', name: 'Brazil' },
                { code: 'COL', name: 'Colombia' },
                { code: 'ARG', name: 'Argentina' }
            ]
        },
        {
            code: 'OC', name: 'Oceania', countries: [
                { code: 'AUS', name: 'Australia' },
                { code: 'NZL', name: 'New Zealand' },
                { code: 'WSM', name: 'Samoa' }
            ]
        },
        {
            code: 'AN', name: 'Antarctica', countries: [
                { code: 'BVT', name: 'Bouvet Island' },
                { code: 'ATF', name: 'French Southern Lands' }
            ]
        },
    ];
    return {
      fields: { dataSource: dataSource, id: 'code', text: 'name', child:
'countries' }
    }
  }
}
</script>
<style>
  @import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 350px;
        max-height: 350px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/data-binding/hierarchical-data-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Self-referential data

TreeView can be populated from self-referential data structure that contains array of JSON objects with parentID mapping.

You can directly assign self-referential data to the dataSource property, and map all the field members with corresponding keys from self-referential data to fields property.

To render the root level nodes, specify the parentID as null or no need to specify the parentID in dataSource.

In the following example, **ID**, **pid**, **hasChild**, and **name** columns from self-referential data have been mapped to **ID parentID**, **hasChildren**, and **text** fields, respectively.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields"></ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
        { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
        { id: 2, pid: 1, name: 'Hot Singles' },
        { id: 3, pid: 1, name: 'Rising Artists' },
        { id: 4, pid: 1, name: 'Live Music' },
        { id: 6, pid: 1, name: 'Best of 2017 So Far' },
        { id: 7, name: 'Sales and Events', hasChild: true },
        { id: 8, pid: 7, name: '100 Albums - $5 Each' },
        { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
        { id: 10, pid: 7, name: 'CD Deals' },
        { id: 11, name: 'Categories', hasChild: true },
        { id: 12, pid: 11, name: 'Songs' },
        { id: 13, pid: 11, name: 'Bestselling Albums' },
        { id: 14, pid: 11, name: 'New Releases' },
        { id: 15, pid: 11, name: 'Bestselling Songs' },
        { id: 16, name: 'MP3 Albums', hasChild: true },
        { id: 17, pid: 16, name: 'Rock' },
        { id: 18, pid: 16, name: 'Gospel' },
        { id: 19, pid: 16, name: 'Latin Music' },
        { id: 20, pid: 16, name: 'Jazz' },
        { id: 21, name: 'More in Music', hasChild: true },
        { id: 22, pid: 21, name: 'Music Trade-In' },
        { id: 23, pid: 21, name: 'Redeem a Gift Card' },
        { id: 24, pid: 21, name: 'Band T-Shirts' },
    ];
    return {
      fields: { dataSource: dataSource, id: 'id', parentID: 'pid', text:
'name', hasChildren: 'hasChild' }
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 350px;
        max-height: 350px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/data-binding/self-referential-data-cs1" %}

Remote data

TreeView can also be populated from a remote data service with the help of DataManager component and Query property.

It supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of DataManager adaptors.

You can assign service data as an instance of DataManager to the dataSource property. To interact with remote data source, you have to provide the endpoint url.

The DataManager that acts as an interface between the service endpoint and the TreeView requires the following information to interact with service endpoint properly.

- DataManager->url: Defines the service endpoint to fetch data.
- DataManager->adaptor: Defines the adaptor option. By default, ODataAdaptor is used for remote binding.

Adaptor is responsible for processing response and request from/to the service endpoint. The @syncfusion/ej2-data package

provides some predefined adaptors designed to interact with service endpoints. They are,

- UrlAdaptor: Used to interact with remote services. This is the base adaptor for all remote based adaptors.
- ODataAdaptor: Used to interact with OData endpoints.
- ODataV4Adaptor: Used to interact with OData V4 endpoints.
- WebApiAdaptor: Used to interact with Web API created under OData standards.
- WebMethodAdaptor: Used to interact with web methods.

In the following example, ODataV4Adaptor is used to fetch data from remote services. The **EmployeeID**, **FirstName**, and **Title**

Copyright © 2001 -2024 Syncfusion Inc.

columns from Employees table have been mapped to **ID**, **text**, and **hasChildren** fields respectively for first level nodes.

The **OrderID**, **EmployeeID**, and **ShipName** columns from orders table have been mapped to **ID**, **parentID**, and **text** fields

respectively for second level nodes.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields"></ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
import { DataManager,Query,ODataV4Adaptor } from "@syncfusion/ej2-data";
export default {
  data () {
    var remotedata = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc',
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
    });
    return {
      fields: { dataSource: remotedata, query: new
Query().from('Employees').select('EmployeeID,FirstName,Title').take(5), id:
'EmployeeID', text: 'FirstName', hasChildren: 'EmployeeID',
        child: { dataSource: remotedata, query: new
Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5), id:
'OrderID', parentID: 'EmployeeID', text: 'ShipName' }
    }
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 350px;
        max-height: 350px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/data-binding/remote-data-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

Check box in Vue Treeview component

The TreeView component allows you to check more than one node in TreeView without affecting the UI's appearance by enabling the showCheckBox property. When this property is enabled, checkboxappears before each TreeView node text.

- If one of the child nodes is not in a checked state, then the parent node will be in an intermediate state.
- If all the child nodes are in checked state, then the parent node's state will also be checked.
- If a parent node is checked, then all the child nodes' state will also be checked.

By default, the checkbox state of parent and child nodes are dependent on each other. If you need independent checked state, you can achieve it using the autoCheck property.

Using the checkedNodes property, you can set the nodes that need to be checked or get the ID of nodes that are currently checked in the TreeView component.

If you need to prevent the node check action for a particular node, the nodeChecking event can be used which is triggered before the TreeView node is checked/unchecked. The nodeChecked event will be triggered when the TreeView node is checked/unchecked successfully.

In the following example, the showCheckBox property is enabled.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields"
:showCheckBox='true'></ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
        { id: 1, name: 'Australia', hasChild: true, expanded: true },
        { id: 2, pid: 1, name: 'New South Wales', isChecked: true },
        { id: 3, pid: 1, name: 'Victoria' },
        { id: 4, pid: 1, name: 'South Australia' },
        { id: 6, pid: 1, name: 'Western Australia', isChecked: true },
        { id: 7, name: 'Brazil', hasChild: true },
        { id: 8, pid: 7, name: 'Paraná' },
        { id: 9, pid: 7, name: 'Ceará' },
        { id: 10, pid: 7, name: 'Acre' },
        { id: 11, name: 'China', hasChild: true },
        { id: 12, pid: 11, name: 'Guangzhou' },
        { id: 13, pid: 11, name: 'Shanghai' },
        { id: 14, pid: 11, name: 'Beijing' },
        { id: 15, pid: 11, name: 'Shantou' },
        { id: 16, name: 'France', hasChild: true },
        { id: 17, pid: 16, name: 'Pays de la Loire' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        { id: 18, pid: 16, name: 'Aquitaine' },
        { id: 19, pid: 16, name: 'Brittany' },
        { id: 20, pid: 16, name: 'Lorraine' },
        { id: 21, name: 'India', hasChild: true },
        { id: 22, pid: 21, name: 'Assam' },
        { id: 23, pid: 21, name: 'Bihar' },
        { id: 24, pid: 21, name: 'Tamil Nadu' },
        { id: 25, pid: 21, name: 'Punjab' }
    ];
    return {
        fields: { dataSource: dataSource, id: 'id', parentID: 'pid', text:
'name', hasChildren: 'hasChild' },
    }
  }
}
</script>
<style>
  @import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 350px;
        max-height: 350px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/checkbox/checkbox-cs1" %}

Checked nodes

You can get or set the checked nodes in TreeView at initial rendering and dynamically by using the checkedNodes

property. It returns the checked nodes' ID as an array.

In the following example, the **New South Wales** and **Western Australia** nodes are checked at initial rendering. If any more nodes are checked, the checked nodes' IDs will be displayed in alert.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields" :showCheckBox='true'
:checkedNodes='checkedNodes' :nodeChecked='nodeChecked'></ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
        { id: 1, name: 'Australia', hasChild: true, expanded: true },
        { id: 2, pid: 1, name: 'New South Wales', isChecked: true },
        { id: 3, pid: 1, name: 'Victoria' },
        { id: 4, pid: 1, name: 'South Australia' },
        { id: 6, pid: 1, name: 'Western Australia', isChecked: true },
        { id: 7, name: 'Brazil', hasChild: true },
        { id: 8, pid: 7, name: 'Paraná' },
        { id: 9, pid: 7, name: 'Ceará' },
        { id: 10, pid: 7, name: 'Acre' },
        { id: 11, name: 'China', hasChild: true },
        { id: 12, pid: 11, name: 'Guangzhou' },
        { id: 13, pid: 11, name: 'Shanghai' },
        { id: 14, pid: 11, name: 'Beijing' },
        { id: 15, pid: 11, name: 'Shantou' },
        { id: 16, name: 'France', hasChild: true },
        { id: 17, pid: 16, name: 'Pays de la Loire' },
        { id: 18, pid: 16, name: 'Aquitaine' },
        { id: 19, pid: 16, name: 'Brittany' },
        { id: 20, pid: 16, name: 'Lorraine' },
        { id: 21, name: 'India', hasChild: true },
        { id: 22, pid: 21, name: 'Assam' },
        { id: 23, pid: 21, name: 'Bihar' },
        { id: 24, pid: 21, name: 'Tamil Nadu' },
        { id: 25, pid: 21, name: 'Punjab' }
    ];
    return {
        fields: { dataSource: dataSource, id: 'id', parentID: 'pid', text:
'name', hasChildren: 'hasChild' },
        checkedNodes: ['2','6']
    }
  },
  methods: {
        nodeChecked: function(args) {
           var treeObj =
document.getElementById('treeview').ej2_instances[0];
           alert("The checked node's id: " + treeObj.checkedNodes); // To
alert the checked node's id.
        }
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
buttons/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 350px;
        max-height: 350px;
        margin: auto;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/checkbox/checked-nodes-cs1" %}

See Also

- How to check/uncheck the checkbox on clicking the tree node text

Node editing in Vue Treeview component

The TreeView allows you to edit nodes by setting the allowEditing property to **true**. To directly edit the nodes in place, **double click** the TreeView node or **select** the node and press **F2** key.

When editing is completed by focus out or by pressing the **Enter** key, the modified node's text saves automatically. If you do not want to save the modified node's text in TreeView node, press **Escape** key. It does not save the edited text to the TreeView node.

Node editing can also be performed programmatically by using the beginEdit method. On passing the node ID or element through this method, the edit textbox will be created for the particular node thus allowing us to edit it.

If you need to validate or prevent editing, the nodeEditing event can be used which is triggered before the TreeView node is renamed. On successfully renaming a node the nodeEdited event will be triggered.

In the following example, the first level node's text cannot be changed, but all other level nodes' text can be changed.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields" :allowEditing='true'
:nodeEditing='editing'></ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
        { id: '01', name: 'Local Disk (C:)', expanded: true,
            subChild: [
                {
                    id: '01-01', name: 'Program Files',
                    subChild: [
                        { id: '01-01-01', name: '7-Zip' },
                        { id: '01-01-02', name: 'Git' },
                        { id: '01-01-03', name: 'IIS Express' },
```

```
                    ]
                },
                {
                    id: '01-02', name: 'Users', expanded: true,
                    subChild: [
                        { id: '01-02-01', name: 'Smith' },
                        { id: '01-02-02', name: 'Public' },
                        { id: '01-02-03', name: 'Admin' },
                    ]
                },
                {
                    id: '01-03', name: 'Windows',
                    subChild: [
                        { id: '01-03-01', name: 'Boot' },
                        { id: '01-03-02', name: 'FileManager' },
                        { id: '01-03-03', name: 'System32' },
                    ]
                },
            ]
        },
        {
            id: '02', name: 'Local Disk (D:)',
            subChild: [
                {
                    id: '02-01', name: 'Personals',
                    subChild: [
                        { id: '02-01-01', name: 'My photo.png' },
                        { id: '02-01-02', name: 'Rental document.docx' },
                        { id: '02-01-03', name: 'Pay slip.pdf' },
                    ]
                },
                {
                    id: '02-02', name: 'Projects',
                    subChild: [
                        { id: '02-02-01', name: 'ASP Application' },
                        { id: '02-02-02', name: 'TypeScript Application' },
                        { id: '02-02-03', name: 'React Application' },
                    ]
                },
                {
                    id: '02-03', name: 'Office',
                    subChild: [
                        { id: '02-03-01', name: 'Work details.docx' },
                        { id: '02-03-02', name: 'Weekly report.docx' },
                        { id: '02-03-03', name: 'Wish list.csv' },
                    ]
                },
            ]
        },
        {
            id: '03', name: 'Local Disk (E:)', icon: 'folder',
            subChild: [
                {
                    id: '03-01', name: 'Pictures',
                    subChild: [
                        { id: '03-01-01', name: 'Wind.jpg' },
                        { id: '03-01-02', name: 'Stone.jpg' },
```

```
                        { id: '03-01-03', name: 'Home.jpg' },
                    ]
                },
                {
                    id: '03-02', name: 'Documents',
                    subChild: [
                    { id: '03-02-01', name: 'Environment Pollution.docx'
},
                    { id: '03-02-02', name: 'Global Warming.ppt' },
                    { id: '03-02-03', name: 'Social Network.pdf' },
                    ]
                },
                {
                    id: '03-03', name: 'Study Materials',
                    subChild: [
                    { id: '03-03-01', name: 'UI-Guide.pdf' },
                    { id: '03-03-02', name: 'Tutorials.zip' },
                    { id: '03-03-03', name: 'TypeScript.7z' },
                    ]
                },
            ]
        }
    ];
    return {
        fields: { dataSource: dataSource, id: 'id', text: 'name', child:
'subChild' },
        checkedNodes: ['2','6']
    }
  },
  methods: {
        editing: function(args) {
           //check whether node is root node or not
            if (args.node.parentNode.parentNode.nodeName !== "LI") {
                args.cancel = true;
            }
        }
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 350px;
        max-height: 350px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/node-editing-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

See Also

- How to validate the text when renaming the tree node
- How to process the tree node operations using context menu

Multiple selection in Vue Treeview component

Selection provides an interactive support and highlights the node that you select. Selection can be done through simple mouse down or keyboard interaction.

The TreeView also supports selection of multiple nodes by setting allowMultiSelection

to **true**.

To multi-select, press and hold **CTRL** key and click the desired nodes. To select range of nodes, press and hold the

SHIFT key and click the nodes.

In the following example, the allowMultiSelection property is enabled.

Multi selection is not applicable through touch interactions.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields"
:allowMultiSelection='true'></ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
        { id: 1, name: 'Australia', hasChild: true, expanded: true },
        { id: 2, pid: 1, name: 'New South Wales', isSelected: true },
        { id: 3, pid: 1, name: 'Victoria' },
        { id: 4, pid: 1, name: 'South Australia' },
        { id: 6, pid: 1, name: 'Western Australia', isSelected: true },
        { id: 7, name: 'Brazil', hasChild: true },
        { id: 8, pid: 7, name: 'Paraná' },
        { id: 9, pid: 7, name: 'Ceará' },
        { id: 10, pid: 7, name: 'Acre' },
        { id: 11, name: 'China', hasChild: true },
        { id: 12, pid: 11, name: 'Guangzhou' },
        { id: 13, pid: 11, name: 'Shanghai' },
        { id: 14, pid: 11, name: 'Beijing' },
        { id: 15, pid: 11, name: 'Shantou' },
        { id: 16, name: 'France', hasChild: true },
        { id: 17, pid: 16, name: 'Pays de la Loire' },
        { id: 18, pid: 16, name: 'Aquitaine' },
        { id: 19, pid: 16, name: 'Brittany' },
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        { id: 20, pid: 16, name: 'Lorraine' },
        { id: 21, name: 'India', hasChild: true },
        { id: 22, pid: 21, name: 'Assam' },
        { id: 23, pid: 21, name: 'Bihar' },
        { id: 24, pid: 21, name: 'Tamil Nadu' },
        { id: 25, pid: 21, name: 'Punjab' }
    ];
    return {
        fields: { dataSource: dataSource, id: 'id', parentID: 'pid', text:
'name', hasChildren: 'hasChild', selected: 'isSelected' },
    }
  }
}
</script>
<style>
  @import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 350px;
        max-height: 350px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/multi-selection/selection-cs1" %}

Selected nodes

You can get or set the selected nodes in TreeView at initial rendering and dynamically by using the selectedNodes property. It will return the selected node's ID as an array.

The nodeselecting event is triggered before a node is selected/unselected which can be used to prevent the selection.

The nodeSelected event is triggered once a node is successfully selected/unselected.

In the following example, **New South Wales** and **Western Australia** nodes are selected at initial rendering. When a node is selected, the selected node's ID is displayed in alert.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields"
:allowMultiSelection='true' :selectedNodes="selectedNodes"
:nodeSelected="nodeSelected"></ejs-treeview>
    </div>
  </div>
</template>
<script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
        { id: 1, name: 'Australia', hasChild: true, expanded: true },
        { id: 2, pid: 1, name: 'New South Wales', isSelected: true },
        { id: 3, pid: 1, name: 'Victoria' },
        { id: 4, pid: 1, name: 'South Australia' },
        { id: 6, pid: 1, name: 'Western Australia', isSelected: true },
        { id: 7, name: 'Brazil', hasChild: true },
        { id: 8, pid: 7, name: 'Paraná' },
        { id: 9, pid: 7, name: 'Ceará' },
        { id: 10, pid: 7, name: 'Acre' },
        { id: 11, name: 'China', hasChild: true },
        { id: 12, pid: 11, name: 'Guangzhou' },
        { id: 13, pid: 11, name: 'Shanghai' },
        { id: 14, pid: 11, name: 'Beijing' },
        { id: 15, pid: 11, name: 'Shantou' },
        { id: 16, name: 'France', hasChild: true },
        { id: 17, pid: 16, name: 'Pays de la Loire' },
        { id: 18, pid: 16, name: 'Aquitaine' },
        { id: 19, pid: 16, name: 'Brittany' },
        { id: 20, pid: 16, name: 'Lorraine' },
        { id: 21, name: 'India', hasChild: true },
        { id: 22, pid: 21, name: 'Assam' },
        { id: 23, pid: 21, name: 'Bihar' },
        { id: 24, pid: 21, name: 'Tamil Nadu' },
        { id: 25, pid: 21, name: 'Punjab' }
    ];
    return {
        fields: { dataSource: dataSource, id: 'id', parentID: 'pid', text:
'name', hasChildren: 'hasChild', selected: 'isSelected' },
        selectedNodes: ['2','6'],
    }
  },
  methods: {
        nodeSelected: function(args) {
           var treeObj =
document.getElementById('treeview').ej2_instances[0];
            alert("The selected node's id: " + treeObj.selectedNodes); // To
alert the selected node's id.
        }
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 350px;
        max-height: 350px;
        margin: auto;
        overflow: auto;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/multi-selection/selection-cs2" %}

Drag and drop in Vue Treeview component

The TreeView component allows you to drag and drop any node by setting allowDragAndDrop to **true**. Nodes can be dragged and dropped at all levels of the same TreeView.

The dragged nodes can be dropped at any level by indicator lines with **plus**, **minus/restrict**, and **in between** icons. It represents the exact position where the node is to be dropped as sibling or child.

If you need to prevent dragging action for a particular node, the nodeDragStart event can be used which is triggered when the node drag is started. If you need to prevent dropping action for a particular node, the nodeDragStop event can be used which is triggered when the drag is stopped.

The nodeDragging event is triggered when the TreeView node is being dragged. You can customize the cloned element in this event.

The nodeDropped event is triggered when the TreeView node is dropped on the target element successfully.

The following table explains the usage of indicator icons.

| Icons | Description |

|------|-------------|

| Plus icon | Indicates that the dragged node is to be added as child of target node. |

| Minus or restrict icon |Indicates that the dragged node is not to be dropped at the hovered region. |

| In between icon | Indicates that the dragged node is to be added as siblings of hovered region. |

In the following sample, the allowDragAndDrop property is enabled.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields"
:allowDragAndDrop='true'></ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
        {
        id: 1, name: 'ASP.NET MVC Team', expanded: true,
        child: [
```

```
            { id: 2, pid: 1, name: 'Smith' },
            { id: 3, pid: 1, name: 'Johnson', isSelected: true },
            { id: 4, pid: 1, name: 'Anderson' },
        ]
    },
    {
        id: 5, name: 'Windows Team',
        child: [
            { id: 6, pid: 5, name: 'Clark' },
            { id: 7, pid: 5, name: 'Wright' },
            { id: 8, pid: 5, name: 'Lopez' },
        ]
    },
    {
        id: 9, name: 'Web Team',
        child: [
            { id: 11, pid: 9, name: 'Joshua' },
            { id: 12, pid: 9, name: 'Matthew' },
            { id: 13, pid: 9, name: 'David' },
        ]
    },
    {
        id: 14, name: 'Build Team',
        child: [
            { id: 15, pid: 14, name: 'Ryan' },
            { id: 16, pid: 14, name: 'Justin' },
            { id: 17, pid: 14, name: 'Robert' },
        ]
    },
    {
        id: 18, name: 'WPF Team',
        child: [
            { id: 19, pid: 18, name: 'Brown' },
            { id: 20, pid: 18, name: 'Johnson' },
            { id: 21, pid: 18, name: 'Miller' },
        ]
    }
    ];
    return {
        fields: { dataSource: dataSource, id: 'id', text: 'name', child:
'child', selected: 'isSelected' },
    }
  }
}
</script>
<style>
  @import
"https://ej2.syncfusion.com/vue/documentation/../node_modules/@syncfusion/ej
2-vue-navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 350px;
        max-height: 350px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        border-radius: 3px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/drag-and-drop/single-cs1" %}

Multiple-node drag and drop

To drag and drop more than one node, you should enable the allowMultiSelection property along with the allowDragAndDrop property.

To perform multi-selection, press and hold **CTRL** key and click the desired nodes. To select range of nodes, press and hold the **SHIFT** key and click the nodes.

In the following sample, the allowMultiSelection property is enabled along with the allowDragAndDrop property.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields"
:allowDragAndDrop='true' :allowMultiSelection='true'></ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
        {
        id: 1, name: 'ASP.NET MVC Team', expanded: true,
        child: [
            { id: 2, pid: 1, name: 'Smith' },
            { id: 3, pid: 1, name: 'Johnson', isSelected: true },
            { id: 4, pid: 1, name: 'Anderson', isSelected: true },
        ]
    },
    {
        id: 5, name: 'Windows Team',
        child: [
            { id: 6, pid: 5, name: 'Clark' },
            { id: 7, pid: 5, name: 'Wright' },
            { id: 8, pid: 5, name: 'Lopez' },
        ]
    },
    {
        id: 9, name: 'Web Team',
        child: [
            { id: 11, pid: 9, name: 'Joshua' },
            { id: 12, pid: 9, name: 'Matthew' },
            { id: 13, pid: 9, name: 'David' },
        ]
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    },
    {
        id: 14, name: 'Build Team',
        child: [
            { id: 15, pid: 14, name: 'Ryan' },
            { id: 16, pid: 14, name: 'Justin' },
            { id: 17, pid: 14, name: 'Robert' },
        ]
    },
    {
        id: 18, name: 'WPF Team',
        child: [
            { id: 19, pid: 18, name: 'Brown' },
            { id: 20, pid: 18, name: 'Johnson' },
            { id: 21, pid: 18, name: 'Miller' },
        ]
    }
    ];
    return {
        fields: { dataSource: dataSource, id: 'id', text: 'name', child:
'child', selected: 'isSelected' },
    }
  }
}
</script>
<style>
   @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
   @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 350px;
        max-height: 350px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/drag-and-drop/multiple-cs1" %}

See Also

- How to restrict the drag-and-drop for particular tree nodes

Template in Vue Treeview component

The TreeView component allows you to customize the look of TreeView nodes by using the nodeTemplate

property. This property accepts either `template string` or HTML element ID.

In the following sample, employee information such as employee photo, name, and designation have been included using the `nodeTemplate` property.

Copyright © 2001 -2024 Syncfusion Inc.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields" cssClass='custom'
:nodeTemplate='Template'></ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var demoVue = Vue.component("demo", {
    template: "<div><img class='eimage' :src='data.eimg'
alt='employee'/><div class='ename'>{{data.name}}</div><div
class='ejob'>{{data.job}}</div></div>"
    data() {
        return {
        data: {}
        };
    }
    });
    var dataSource =  [
        { id: 1, name: 'Steven Buchanan', eimg:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/10.png',
job: 'CEO', hasChild: true, expanded: true },
        { id: 2, pid: 1, name: 'Laura Callahan', eimg:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/2.png', job:
'Product Manager', hasChild: true },
        { id: 3, pid: 2, name: 'Andrew Fuller', eimg:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/7.png', job:
'Team Lead', hasChild: true },
        { id: 4, pid: 3, name: 'Anne Dodsworth', eimg:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/1.png', job:
'Developer' },
        { id: 5, pid: 1, name: 'Nancy Davolio', eimg:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/4.png', job:
'Product Manager', hasChild: true },
        { id: 6, pid: 5, name: 'Michael Suyama', eimg:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/9.png', job:
'Team Lead', hasChild: true },
        { id: 7, pid: 6, name: 'Robert King', eimg:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/8.png', job:
'Developer ' },
        { id: 8, pid: 7, name: 'Margaret Peacock', eimg:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/6.png', job:
'Developer' },
        { id: 9, pid: 1, name: 'Janet Leverling', eimg:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/3.png', job:
'HR' },
    ];
    return {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        fields: { dataSource: dataSource, id: 'id', parentID: 'pid', text:
'name', hasChildren: 'hasChild' },
        Template: function(e) {
            return {
                template: demoVue
            };
        }
    }
  }}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 450px;
        max-height: 350px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
    .custom .e-list-item .e-fullrow {
        height: 72px;
    }
    .custom .e-list-item .e-list-text {
        line-height: normal;
    }
    .eimage {
        float: left;
        padding: 11px 16px 11px 0;
    }
    .ename {
        font-size: 16px;
        padding: 14px 0 0;
    }
    .ejob {
        font-size: 14px;
        opacity: .87;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/template-cs1" %}

See Also

- How to customize the expand and collapse icons
- How to customize the tree nodes based on levels

Accessibility in Vue Treeview component

The TreeView component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the TreeView component is outlined below.

Copyright © 2001 -2024 Syncfusion Inc.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

WAI-ARIA attributes

The TreeView component followed the WAI-ARIA patterns to meet the accessibility. The following ARIA attributes are used in the TreeView component:

| Attributes | Purpose |

| --- | --- |

| role=tree | All tree nodes are contained within the element. |

| role=treeitem | Specifies the role of each tree node in a selectable TreeView and its containment within the tree. |

| role=group | Specifies the role of each parent node container. |

| role=checkbox | Indicates checkbox control along with treeitem element. |

| aria-multiselectable | Indicates whether the TreeView enables multiple selection or not. |

| aria-expanded | Indicates whether the parent node has expanded or not. |

| aria-selected | Indicates the selected node. |

| aria-grabbed | Indicates the selected state on drag-and-drop of node. |

| aria-level | Indicates the level of node in TreeView. |

| aria-checked | Indicates the current checked state of TreeView checkbox. |

| aria-label | Indicates the contextual message for the TreeView checkbox. |

| aria-activedescendant | Identifies the currently active element when focusing on the TreeView. |

| aria-disabled | Indicates element is perceivable but disabled. |

Keyboard interaction

The TreeView component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the TreeView component.

| Interaction Keys | Description |

|------|---------|

| Arrow Up | Goes to the previous node. |

| Arrow Down | Goes to the next node. |

| Arrow Right | Expands the current node. |

| Arrow Left | Collapses the current node. |

| Home | Goes to the first node. |

| End | Goes to the last node. |

| F2 | Edits the focused node. |

| Esc | Focuses out the edit state without saving the edited text. |

| Enter | Selects the focused node/saves the edited text. |

| Space | Checks the current node. |

| Ctrl + A | Selects all nodes. |

Copyright © 2001 -2024 Syncfusion Inc.

Ensuring accessibility

The TreeView component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the TreeView component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the TreeView component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/treeview.html" %}

See also

- Accessibility in Syncfusion Vue components

How To

Customize the expand and collapse icons in Vue Treeview component

You can customize TreeView expand and collapse icons by using the cssClass property of TreeView.

Refer to the sample to customize expand/collapse icons.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields"
cssClass='custom'></ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
        { id: '01', name: 'Local Disk (C:)', expanded: true,
        subChild: [
            {
                id: '01-01', name: 'Program Files',
                subChild: [
                    { id: '01-01-01', name: '7-Zip' },
                    { id: '01-01-02', name: 'Git' },
                    { id: '01-01-03', name: 'IIS Express' },
                ]
            },
            {
                id: '01-02', name: 'Users', expanded: true,
                subChild: [
                    { id: '01-02-01', name: 'Smith' },
                    { id: '01-02-02', name: 'Public' },
                    { id: '01-02-03', name: 'Admin' },
                ]
```

```
            },
            {
                id: '01-03', name: 'Windows',
                subChild: [
                    { id: '01-03-01', name: 'Boot' },
                    { id: '01-03-02', name: 'FileManager' },
                    { id: '01-03-03', name: 'System32' },
                ]
            },
        ]
    },
    {
        id: '02', name: 'Local Disk (D:)',
        subChild: [
            {
                id: '02-01', name: 'Personals',
                subChild: [
                    { id: '02-01-01', name: 'My photo.png' },
                    { id: '02-01-02', name: 'Rental document.docx' },
                    { id: '02-01-03', name: 'Pay slip.pdf' },
                ]
            },
            {
                id: '02-02', name: 'Projects',
                subChild: [
                    { id: '02-02-01', name: 'ASP Application' },
                    { id: '02-02-02', name: 'TypeScript Application' },
                    { id: '02-02-03', name: 'React Application' },
                ]
            },
            {
                id: '02-03', name: 'Office',
                subChild: [
                    { id: '02-03-01', name: 'Work details.docx' },
                    { id: '02-03-02', name: 'Weekly report.docx' },
                    { id: '02-03-03', name: 'Wish list.csv' },
                ]
            },
        ]
    },
    {
        id: '03', name: 'Local Disk (E:)', icon: 'folder',
        subChild: [
            {
                id: '03-01', name: 'Pictures',
                subChild: [
                    { id: '03-01-01', name: 'Wind.jpg' },
                    { id: '03-01-02', name: 'Stone.jpg' },
                    { id: '03-01-03', name: 'Home.jpg' },
                ]
            },
            {
                id: '03-02', name: 'Documents',
                    subChild: [
                    { id: '03-02-01', name: 'Environment Pollution.docx' },
                    { id: '03-02-02', name: 'Global Warming.ppt' },
                    { id: '03-02-03', name: 'Social Network.pdf' },
```

```
                ]
            },
            {
                id: '03-03', name: 'Study Materials',
                subChild: [
                    { id: '03-03-01', name: 'UI-Guide.pdf' },
                    { id: '03-03-02', name: 'Tutorials.zip' },
                    { id: '03-03-03', name: 'TypeScript.7z' },
                ]
            },
            ]
        }
    ];
    return {
        fields: { dataSource: dataSource, id: 'id', text: 'name', child:
'subChild' },
    }
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 450px;
        max-height: 350px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
.custom .e-list-item .e-icons {
    font-family: "Customize-icon";
  }

  .custom.e-treeview .e-list-item .e-icon-expandable::before, .custom.e-
treeview .e-list-item .e-icon-collapsible:before {
    content: '\e700';
    font-size: 12px;
  }

  @font-face {
    font-family: 'Customize-icon';
    src: url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj0gSRcAAAEoAAAAVmNtYXDnEOdaAAABjAAAADhnbHlmcYq
IngAAAcwAAAD8aGVhZBWT124AAADQAAAANmhoZWEHmANtAAAArAAAACRobXR4C9AAAAAAAYAAAAA
MbG9jYQBAAH4AAAHEAAAACG1heHABEAAxAAABCAAAACBuYW1l/qscPAAAAsgAAAJ5cG9zdIPGFvo
AAAVEAAAAVgABAAADUv9qAFoEAAAA//8D6QABAAAAAAAAAAAAAAAAAAAAAwABAAAAAQAAIKcGUl8
PPPUACwPoAAAAANlGSVAAAAAA2UZJUAAAAAAD6QPpAAAACAACAAAAAAAAAAEAAAADACUAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPwAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAQNS/2oAWgPpAJYAAAABAAAAAAAABAAAAAPoAAA
D6AAAAAAAAgAAAAMAAAAUAAMAAQAAABQABAAkAAAABAAEAAEAAOcB//8AAOcA//8AAAABAAQAAAA
BAAIAAAAAAEAAfgADAAAAAAPpA+kACAAWACQAAAEhFSEHMzcnIyUWEAcGICcmEDc+ATIWBQYQFxY
gNzYQJy4BIgYCMf6kAWqUqMK8rgF+goKK/qCEfn5Coquf/amRkZoBkpqRkUq3xLcCKmSTybt4if6
ghYKChQFgiUJBQRma/m6akZGaAZKaSElJAAMAAAAAA+gD6QAGABQAIgAAASMXNyMRIyUWEAcGICc
```

```
mEDc+ATIWBQYQFxYgNzYQJy4BIgYBvrLp6JmGAW6BgYf+oYiBgUGhqqH9qZOTmgGOmpOTSrbCtgG
y6ekBbwuI/qGHgYGIAV6IQEFBFpr+cZmTk5oBj5lKSUkAAAAAABIA3gABAAAAAAAAAAAEAAAABAAA
AAAABAA4AAQABAAAAAAACAAcADwABAAAAAAADAA4AFgABAAAAAAAAEAA4AJAABAAAAAAAFAAsAMgA
BAAAAAAAGAA4APQABAAAAAAAKACwASwABAAAAAAALABIAdwADAAEECQAAAAIAiQADAAEECQABABw
AiwADAAEECQACAA4ApwADAAEECQADABwAtQADAAEECQAEABwA0QADAAEECQAFABYA7QADAAEECQA
GABwBAwADAAEECQAKAFgBHwADAAEECQALACQBdyBDdXN0b21pemUtaWNvblJlZ3VsYXJDdXN0b21
pemUtaWNvbkN1c3RvbWl6ZS1pY29uVmVyc2lvbiAxLjBDdXN0b21pemUtaWNvbkZvbnQgZ2VuZXJ
hdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAQwB
1AHMAdABvAG0AaQB6AGUALQBpAGMAbwBuAFIAZQBnAHUAbABhAHIAQwB1AHMAdABvAG0AaQB6AGU
ALQBpAGMAbwBuAEMAdQBzAHQAbwBtAGkAegBlAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQA
uADAAQwB1AHMAdABvAG0AaQB6AGUALQBpAGMAbwBuAEYAbwBuAHQAIABnAGUAbgBlAHIAYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdABhAG8AIABTAHQAdQB
kAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAMBAgEDAQQAFy1hcnJvdy1jaXJjbGUtcmlnaHQtXzAxEi1hcnJ
vdy1jaXJjbGUtZG93bgAAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
  }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/how-to/icon-cs1" %}

Process the tree node operations using context menu in Vue Treeview component

You can integrate the context menu with TreeView component in order to perform the tree view related operations like add, remove and renaming node. Following is an example which demonstrates the above cases which are used to manipulate tree view operations in the 'select ' event of context menu.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields"
:nodeClicked='nodeclicked'>
            <ejs-contextmenu id='contentmenutree' target='#treeview'
:items='menuItems' :beforeOpen='beforeopen' :select='menuclick'></ejs-
contextmenu>
        </ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { ContextMenuPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(ContextMenuPlugin);
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
      var dataSource =  [
        { id: '01', name: 'Local Disk (C:)', expanded: true,
hasAttribute:{class:'remove rename'},
            subChild: [
                {
                    id: '01-01', name: 'Program Files',
                    subChild: [
```

```
                        { id: '01-01-01', name: 'Windows NT' },
                        { id: '01-01-02', name: 'Windows Mail' },
                        { id: '01-01-03', name: 'Windows Photo Viewer' },
                    ]
                },
                {
                    id: '01-02', name: 'Users', expanded: true,
                    subChild: [
                        { id: '01-02-01', name: 'Smith' },
                        { id: '01-02-02', name: 'Public' },
                        { id: '01-02-03', name: 'Admin' },
                    ]
                },
                {
                    id: '01-03', name: 'Windows',
                    subChild: [
                        { id: '01-03-01', name: 'Boot' },
                        { id: '01-03-02', name: 'FileManager' },
                        { id: '01-03-03', name: 'System32' },
                    ]
                },
            ]
        },
        {
            id: '02', name: 'Local Disk (D:)',
hasAttribute:{class:'rename'},
            subChild: [
                {
                    id: '02-01', name: 'Personals',
                    subChild: [
                        { id: '02-01-01', name: 'My photo.png' },
                        { id: '02-01-02', name: 'Rental document.docx' },
                        { id: '02-01-03', name: 'Pay slip.pdf' },
                    ]
                },
                {
                    id: '02-02', name: 'Projects',
                    subChild: [
                        { id: '02-02-01', name: 'ASP Application' },
                        { id: '02-02-02', name: 'TypeScript Application' },
                        { id: '02-02-03', name: 'React Application' },
                    ]
                },
                {
                    id: '02-03', name: 'Office',
                    subChild: [
                        { id: '02-03-01', name: 'Work details.docx' },
                        { id: '02-03-02', name: 'Weekly report.docx' },
                        { id: '02-03-03', name: 'Wish list.csv' },
                    ]
                },
            ]
        },
        {
            id: '03', name: 'Local Disk (E:)', icon: 'folder',
hasAttribute:{class:'remove'},
            subChild: [
```

```
                {
                    id: '03-01', name: 'Pictures',
                    subChild: [
                        { id: '03-01-01', name: 'Wind.jpg' },
                        { id: '03-01-02', name: 'Stone.jpg' },
                        { id: '03-01-03', name: 'Home.jpg' },
                    ]
                },
                {
                    id: '03-02', name: 'Documents',
                    subChild: [
                        { id: '03-02-01', name: 'Environment Pollution.docx'
},
                        { id: '03-02-02', name: 'Global Warming.ppt' },
                        { id: '03-02-03', name: 'Social Network.pdf' },
                    ]
                },
                {
                    id: '03-03', name: 'Study Materials',
                    subChild: [
                        { id: '03-03-01', name: 'UI-Guide.pdf' },
                        { id: '03-03-02', name: 'Tutorials.zip' },
                        { id: '03-03-03', name: 'TypeScript.7z' },
                    ]
                },
            ]
        }
    ];
    return {
        fields : { dataSource: dataSource, id: 'id', text: 'name', child:
'subChild', htmlAttributes: 'hasAttribute' },
        menuItems : [
            { text: 'Add New Item' },
            { text: 'Rename Item' },
            { text: 'Remove Item' }
        ],
        index : 1,
    }
  },
   methods: {
       nodeclicked: function(args) {
            if (args.event.which === 3) {
                var treeObj =
document.getElementById('treeview').ej2_instances[0];
                treeObj.selectedNodes = [args.node.getAttribute('data-
uid')];
            }
        }
        menuclick: function(args) {
            var treevalidate =
document.getElementById('treeview').ej2_instances[0];
            var targetNodeId = treevalidate.selectedNodes[0];
            if (args.item.text == "Add New Item") {
                var nodeId = "tree_" + this.index;
                var item = { id: nodeId, name: "New Folder" };
                treevalidate.addNodes([item], targetNodeId, null);
                this.index++;
```

```
                treevalidate.fields.dataSource.push(item);
                treevalidate.beginEdit(nodeId);
            }
            else if (args.item.text == "Remove Item") {
                treevalidate.removeNodes([targetNodeId]);
            }
            else if (args.item.text == "Rename Item") {
                treevalidate.beginEdit(targetNodeId);
            }
        },
        beforeopen: function(args) {
            var treevalidate =
document.getElementById('treeview').ej2_instances[0];
            var targetNodeId = treevalidate.selectedNodes[0];
            var targetNode = document.querySelector('[data-uid="' +
targetNodeId + '"]');
            var contentmenutree =
document.getElementById('contentmenutree').ej2_instances[0];
            if (targetNode.classList.contains('remove')) {
                contentmenutree.enableItems(['Remove Item'], false);
            }
            else {
                contentmenutree.enableItems(['Remove Item'], true);
            }
            if (targetNode.classList.contains('rename')) {
                contentmenutree.enableItems(['Rename Item'], false);
            }
            else {
                contentmenutree.enableItems(['Rename Item'], true);
            }
        }
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 400px;
        max-height: 320px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
    #app {
        width: 350px;
        margin: 0 auto;
    }
     #contentmenutree {
         padding: 0px;
         font-size: inherit;
    }
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/treeview/how-to/context-menu-cs1" %}

Check uncheck the checkbox on clicking the tree node text in Vue Treeview component

You can check and uncheck the checkboxes of tree view by clicking the tree node using the nodeClicked event of TreeView.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields" :showCheckBox='true'
:nodeClicked='nodeChecked' :keyPress='nodeChecked'></ejs-treeview>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
        { id: 1, name: 'Australia', hasChild: true, expanded: true },
        { id: 2, pid: 1, name: 'New South Wales', isChecked: true },
        { id: 3, pid: 1, name: 'Victoria' },
        { id: 4, pid: 1, name: 'South Australia' },
        { id: 6, pid: 1, name: 'Western Australia', isChecked: true },
        { id: 7, name: 'Brazil', hasChild: true },
        { id: 8, pid: 7, name: 'Paraná' },
        { id: 9, pid: 7, name: 'Ceará' },
        { id: 10, pid: 7, name: 'Acre' },
        { id: 11, name: 'China', hasChild: true },
        { id: 12, pid: 11, name: 'Guangzhou' },
        { id: 13, pid: 11, name: 'Shanghai' },
        { id: 14, pid: 11, name: 'Beijing' },
        { id: 15, pid: 11, name: 'Shantou' },
        { id: 16, name: 'France', hasChild: true },
        { id: 17, pid: 16, name: 'Pays de la Loire' },
        { id: 18, pid: 16, name: 'Aquitaine' },
        { id: 19, pid: 16, name: 'Brittany' },
        { id: 20, pid: 16, name: 'Lorraine' },
        { id: 21, name: 'India', hasChild: true },
        { id: 22, pid: 21, name: 'Assam' },
        { id: 23, pid: 21, name: 'Bihar' },
        { id: 24, pid: 21, name: 'Tamil Nadu' },
        { id: 25, pid: 21, name: 'Punjab' }
    ];
    return {
        fields: { dataSource: dataSource, id: 'id', parentID: 'pid', text:
'name', hasChildren: 'hasChild' },
    }
  },
  methods: {
        nodeChecked: function(args) {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            var treeObj =
document.getElementById('treeview').ej2_instances[0];
            var checkedNode = [args.node];
             if (args.event.target.classList.contains('e-fullrow') ||
args.event.key == "Enter") {
                 var getNodeDetails = treeObj.getNode(args.node);
                 if (getNodeDetails.isChecked == 'true') {
                     treeObj.uncheckAll(checkedNode);
                 } else {
                     treeObj.checkAll(checkedNode);
                 }
             }
         }
     }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 350px;
        max-height: 350px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/how-to/checkbox-cs1" %}

Validate the text when renaming the tree node in Vue Treeview component

You can validate the tree node text while editing using `nodeEdited` event of the TreeView. Following is an example that shows how to validate and prevent empty values in tree node.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields" :allowEditing='true'
:nodeEdited='nodeEdited'></ejs-treeview>
    </div>
    <div id="display"></div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
```

```
        {
            id: 1, name: 'Discover Music', expanded: true,
            child: [
                { id: 2, name: 'Hot Singles' },
                { id: 3, name: 'Rising Artists' },
                { id: 4, name: 'Live Music' }
            ]
        },
        {
            id: 7, name: 'Sales and Events',
            child: [
                { id: 8, name: '100 Albums - $5 Each' },
                { id: 9, name: 'Hip-Hop and R&B Sale' },
                { id: 10, name: 'CD Deals' }
            ]
        },
        {
            id: 11, name: 'Categories',
            child: [
                { id: 12, name: 'Songs' },
                { id: 13, name: 'Bestselling Albums' },
                { id: 14, name: 'New Releases' },
                { id: 15, name: 'Bestselling Songs' }
            ]
        },
        {
            id: 16, name: 'MP3 Albums',
            child: [
                { id: 17, name: 'Rock' },
                { id: 18, name: 'Gospel' },
                { id: 19, name: 'Latin Music' },
                { id: 20, name: 'Jazz' }
            ]
        },
        {
            id: 21, name: 'More in Music',
            child: [
                { id: 22, name: 'Music Trade-In' },
                { id: 23, name: 'Redeem a Gift Card' },
                { id: 24, name: 'Band T-Shirts' }
            ]
        }
    ];
    return {
        fields: { dataSource: dataSource, id: 'id', text: 'name', child:
'child' },
    }
  },
  methods: {
        nodeEdited: function(args) {
            let displayContent:string = "";
            if (args.newText.trim() == "") {
                args.cancel = true;
                displayContent = "TreeView item text should not be empty";
            }
            else if (args.newText != args.oldText) {
                displayContent = "TreeView item text edited successfully";
```

```
                } else {
                    displayContent = "";
                }
                document.getElementById("display").innerHTML =
displayContent;
            }
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-inputs/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 400px;
        max-height: 320px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
    #display {
        max-width: 500px;
        margin: auto;
        padding: 10px;
    }
    #app {
        width: 350px;
        margin: 0 auto;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/how-to/validate-cs1" %}

Customize the tree nodes based on levels in Vue Treeview component

You can customize the tree nodes level wise by adding custom cssClass to the component and enabling styles.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields" :allowEditing='true'
cssClass='mytree'></ejs-treeview>
    </div>
    <div class="details">
        <label>Note:</label>
        <div><b>1. The font-weight "Bold" is applied for all the leaf
nodes</b></div>
        <div><i>2. The font-weight "Italic" is applied for first level
nodes</i></div>
        <div style="color: darkmagenta">3. The color "darkmagenta" is
applied for second level nodes</div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
    var dataSource =  [
        { id: '01', name: 'Local Disk (C:)', expanded: true,
            subChild: [
                {
                    id: '01-01', name: 'Program Files',
                    subChild: [
                        { id: '01-01-01', name: 'Windows NT' },
                        { id: '01-01-02', name: 'Windows Mail' },
                        { id: '01-01-03', name: 'Windows Photo Viewer' },
                    ]
                },
                {
                    id: '01-02', name: 'Users', expanded: true,
                    subChild: [
                        { id: '01-02-01', name: 'Smith' },
                        { id: '01-02-02', name: 'Public' },
                        { id: '01-02-03', name: 'Admin' },
                    ]
                },
                {
                    id: '01-03', name: 'Windows',
                    subChild: [
                        { id: '01-03-01', name: 'Boot' },
                        { id: '01-03-02', name: 'FileManager' },
                        { id: '01-03-03', name: 'System32' },
                    ]
                },
            ]
        },
        {
            id: '02', name: 'Local Disk (D:)',
            subChild: [
                {
                    id: '02-01', name: 'Personals',
                    subChild: [
                        { id: '02-01-01', name: 'My photo.png' },
                        { id: '02-01-02', name: 'Rental document.docx' },
                        { id: '02-01-03', name: 'Pay slip.pdf' },
                    ]
                },
                {
                    id: '02-02', name: 'Projects',
                    subChild: [
                        { id: '02-02-01', name: 'ASP Application' },
                        { id: '02-02-02', name: 'TypeScript Application' },
                        { id: '02-02-03', name: 'React Application' },
                    ]
                },
```

```
                {
                    id: '02-03', name: 'Office',
                    subChild: [
                        { id: '02-03-01', name: 'Work details.docx' },
                        { id: '02-03-02', name: 'Weekly report.docx' },
                        { id: '02-03-03', name: 'Wish list.csv' },
                    ]
                },
            ]
        },
        {
            id: '03', name: 'Local Disk (E:)', icon: 'folder',
            subChild: [
                {
                    id: '03-01', name: 'Pictures',
                    subChild: [
                        { id: '03-01-01', name: 'Wind.jpg' },
                        { id: '03-01-02', name: 'Stone.jpg' },
                        { id: '03-01-03', name: 'Home.jpg' },
                    ]
                },
                {
                    id: '03-02', name: 'Documents',
                    subChild: [
                        { id: '03-02-01', name: 'Environment Pollution.docx'
},
                        { id: '03-02-02', name: 'Global Warming.ppt' },
                        { id: '03-02-03', name: 'Social Network.pdf' },
                    ]
                },
                {
                    id: '03-03', name: 'Study Materials',
                    subChild: [
                        { id: '03-03-01', name: 'UI-Guide.pdf' },
                        { id: '03-03-02', name: 'Tutorials.zip' },
                        { id: '03-03-03', name: 'TypeScript.7z' },
                    ]
                },
            ]
        }
    ];
    return {
        fields: { dataSource: dataSource, id: 'id', text: 'name', child:
'subChild' },
    }
  }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 400px;
        max-height: 350px;
        margin: auto;
```

```
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
    #display {
        max-width: 500px;
        margin: auto;
        padding: 10px;
    }
    #app {
        width: 350px;
        margin: 0 auto;
    }
    .details {
    padding-left: 10px;
}
/*apply custom css to first level*/
.mytree .e-level-1 > .e-text-content .e-list-text {
    font-style: italic;
}
/*apply custom css to second level*/
.mytree .e-level-2 > .e-text-content .e-list-text {
    color: darkmagenta;
}
/*apply custom css to all the leaf nodes*/
.mytree .e-level-3 > .e-text-content .e-list-text {
    font-weight: bold;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/how-to/cssClass-cs1" %}

Restrict the drag and drop for particular tree nodes in Vue Treeview component

You can able to restrict to drag and drop files under folder only. These can be achieved by using `nodeDragStop` and `nodeDragging` event of TreeView.

APP.VUE

```
<template>
  <div id="app">
    <div class="control_wrapper">
        <ejs-treeview id='treeview' :fields="fields"
:allowDragAndDrop='true' :nodeDragStop='dragStop'
:nodeDragging='nodeDrag'></ejs-treeview>
    </div>
    <div id="display"></div>
  </div>
</template>
<script>
import Vue from 'vue';
import { TreeViewPlugin } from "@syncfusion/ej2-vue-navigations";
Vue.use(TreeViewPlugin);
export default {
  data () {
         var dataSource =  [
        {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            nodeId: '01', nodeText: 'Music', icon: 'folder',
            nodeChild: [
                { nodeId: '01-01', nodeText: 'Gouttes.mp3', icon: 'audio' }
            ]
        },
        {
            nodeId: '02', nodeText: 'Videos', icon: 'folder',
            nodeChild: [
                { nodeId: '02-01', nodeText: 'Naturals.mp4', icon: 'video'
},
                { nodeId: '02-02', nodeText: 'Wild.mpeg', icon: 'video' },
            ]
        },
        {
            nodeId: '03', nodeText: 'Documents', icon: 'folder',
            nodeChild: [
                { nodeId: '03-01', nodeText: 'Environment Pollution.docx',
icon: 'docx' },
                { nodeId: '03-02', nodeText: 'Global Water, Sanitation, &
Hygiene.docx', icon: 'docx' },
                { nodeId: '03-03', nodeText: 'Global Warming.ppt', icon:
'ppt' },
                { nodeId: '03-04', nodeText: 'Social Network.pdf', icon:
'pdf' },
                { nodeId: '03-05', nodeText: 'Youth Empowerment.pdf', icon:
'pdf' },
            ]
        },
        {
            nodeId: '04', nodeText: 'Pictures', icon: 'folder', expanded:
true,
            nodeChild: [
                {
                    nodeId: '04-01', nodeText: 'Camera Roll', icon:
'folder', expanded: true,
                    nodeChild: [
                        { nodeId: '04-01-01', nodeText:
'WIN_20160726_094117.JPG', image:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/9.png' },
                        { nodeId: '04-01-02', nodeText:
'WIN_20160726_094118.JPG', image:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/3.png' },
                    ]
                },
                { nodeId: '04-02', nodeText: 'Wind.jpg', icon: 'images' },
                { nodeId: '04-03', nodeText: 'Stone.jpg', icon: 'images' },
            ]
        },
        {
            nodeId: '05', nodeText: 'Downloads', icon: 'folder',
            nodeChild: [
                { nodeId: '05-01', nodeText: 'UI-Guide.pdf', icon: 'pdf' },
                { nodeId: '05-02', nodeText: 'Tutorials.zip', icon: 'zip' },
                { nodeId: '05-03', nodeText: 'Game.exe', icon: 'exe' },
                { nodeId: '05-04', nodeText: 'TypeScript.7z', icon: 'zip' },
            ]
        },
```

```
    ];
    return {
        fields: { dataSource: dataSource, id: 'nodeId', text: 'nodeText',
child: 'nodeChild', iconCss: 'icon', imageUrl: 'image' },
    }
  },
  methods: {
      nodeDrag: function(args) {
          if (args.droppedNode != null &&
args.droppedNode.getElementsByClassName('folder') &&
args.droppedNode.getElementsByClassName('folder').length === 0) {
            args.dropIndicator = 'e-no-drop';
          }
      }
      dragStop: function(args) {
         if (args.droppedNode != null &&
args.droppedNode.getElementsByClassName('folder') &&
args.droppedNode.getElementsByClassName('folder').length === 0) {
            args.cancel = true;
          }
    }
}
</script>
<style>
  @import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
  @import "../../node_modules/@syncfusion/ej2-vue-
navigations/styles/material.css";
 .control_wrapper {
        display: block;
        max-width: 400px;
        max-height: 320px;
        margin: auto;
        overflow: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
.e-treeview .e-list-img {
        width: 25px;
        height: 25px;
    }
 /* Loading sprite image for TreeView */
 .e-treeview .e-list-icon {
         background-repeat: no-repeat;
         background-image:
url(https://ej2.syncfusion.com/demos/src/treeview/images/icons/file_icons.pn
g);
         height: 20px;
    }
 /* Specify the icon positions based upon class name */
 .e-treeview .e-list-icon.folder {
         background-position: -10px -552px;
    }
 .e-treeview .e-list-icon.docx {
         background-position: -10px -20px;
     }
 .e-treeview .e-list-icon.ppt {
         background-position: -10px -48px;
```

```
      }
  .e-treeview .e-list-icon.pdf {
          background-position: -10px -104px;
      }
.e-treeview .e-list-icon.images {
          background-position: -10px -132px;
      }
.e-treeview .e-list-icon.zip {
          background-position: -10px -188px;
      }
.e-treeview .e-list-icon.audio {
          background-position: -10px -244px;
      }
.e-treeview .e-list-icon.video {
          background-position: -10px -272px;
      }
.e-treeview .e-list-icon.exe {
          background-position: -10px -412px;
      }
</style>
```

{% previewsample "page.domainurl/code-snippet/treeview/how-to/restrict-drag-drop-cs1" %}

Uploader

Getting Started with the Vue Uploader Component in Vue 2

This article provides a step-by-step guide for setting up a Vue 2 project using Vue-CLI and integrating the Syncfusion Vue Uploader component using the Composition API / Options API.

Prerequisites

System requirements for Syncfusion Vue UI components

Dependencies

The following are the dependencies required to use the uploader component in your application:

`js

|-- @syncfusion/ej2-vue-inputs

|-- @syncfusion/ej2-inputs

|-- @syncfusion/ej2-vue-base

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-buttons

`

Setting up the Vue 2 project

To generate a Vue 2 project using Vue-CLI, use the vue create command. Follow these steps to install Vue CLI and create a new project:

`bash

npm install -g @vue/cli

vue create quickstart

cd quickstart

npm run serve

`

or

`bash

yarn global add @vue/cli

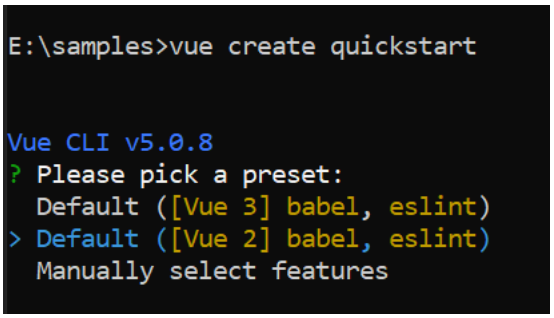
vue create quickstart

cd quickstart

yarn run serve

`

When creating a new project, choose the option Default ([Vue 2] babel, eslint) from the menu.



Once the quickstart project is set up with default settings, proceed to add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion packages are available at npmjs.com. To use Vue components, install the required npm package.

This article uses the Vue Uploader component as an example. Install the @syncfusion/ej2-vue-inputs package by running the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

Copyright © 2001 -2024 Syncfusion Inc.

In this article, the Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Uploader component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

The Custom Resource Generator (CRG) is an online web tool, which can be used to generate the custom script and styles for a set of specific components.

This web tool is useful to combine the required component scripts and styles in a single file.

Add Syncfusion Vue component

Follow the below steps to add the Vue Uploader component using Composition API or Options API:

1\. First, import and register the Uploader component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { UploaderComponent as EjsUploader } from '@syncfusion/ej2-vue-
inputs';
</script>
```

OPTIONS API (~SRC/APP.VUE)

```
<script>
import { UploaderComponent } from '@syncfusion/ej2-vue-inputs';
export default {
components: {
'ejs-uploader': UploaderComponent
}
}
</script>
```

2\. In the template section define the Uploader component.

~/SRC/APP.VUE

```
<template>
<div id="app">
<ejs-uploader ref="uploadObj" id='defaultfileupload'
name="UploadFiles"></ejs-uploader>
</div>
</template>
```

Here is the summarized code for the above steps in the **src/App.vue** file:

Copyright © 2001 -2024 Syncfusion Inc.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div>
    <div id="modalTarget" class="control-section; position:relative"
style="height:350px;">
        <ejs-uploader ref="uploadObj" id='defaultfileupload'
name="UploadFiles"></ejs-uploader>
    </div>
  </div>
</template>
<script setup>
import { UploaderComponent as EjsUploader } from '@syncfusion/ej2-vue-
inputs';
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .control-section {
        height: 100%;
        min-height: 200px;
    }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div>
    <div id="modalTarget" class="control-section; position:relative"
style="height:350px;">
        <ejs-uploader ref="uploadObj" id='defaultfileupload'
name="UploadFiles"></ejs-uploader>
    </div>
  </div>
</template>
<script>
import { UploaderComponent } from '@syncfusion/ej2-vue-inputs';
export default {
    components: {
        'ejs-uploader': UploaderComponent
    },
    data: function() {
        return {  }
    }
}
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #app {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .control-section {
        height: 100%;
        min-height: 200px;
    }
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run serve

`

or

`bash

yarn run serve

`

The Essential JS2 AJAX library has been integrated for uploader server requests. Hence, use the third party promise library like blue-bird to use the uploader in Internet Explorer.

{% previewsample "page.domainurl/code-snippet/uploader/getting-started-cs1" %}

Adding drop area

By default, the uploader component allows to upload files by drag the files from file explorer, and drop into the drop area. You can configure any other external element as drop target using dropArea property.

In the following sample, drop target is configured.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div>
    <div id='droparea'  >
            Drop files here to upload
    </div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:dropArea = "dropElement"></ejs-uploader>
  </div>
</template>
<script setup>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { UploaderComponent as EjsUploader } from '@syncfusion/ej2-vue-
inputs';
const dropElement = '#droparea';
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .fileupload {
        margin: 20px auto;
        width: 400px;
    }
    #droparea {
        padding: 50px 25px;
        margin: 30px auto;
        border: 1px solid #c3c3c3;
        text-align: center;
        width: 20%;
        display: inline-flex;
    }
    .e-file-select,
    .e-file-drop {
        display: none;
    }
    body .e-upload-drag-hover {
         outline: 2px dashed brown;
    }
    #uploadfile {
        width: 60%;
        display: inline-flex;
        margin-left: 5%;
    }
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div>
    <div id='droparea'  >
            Drop files here to upload
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:dropArea = "dropElement"></ejs-uploader>
  </div>
</template>
<script>
import { UploaderComponent } from '@syncfusion/ej2-vue-inputs';
export default {
    components: {
      'ejs-uploader': UploaderComponent
    },
    data: function() {
        return {
        dropElement: '#droparea'
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .fileupload {
        margin: 20px auto;
        width: 400px;
    }
    #droparea {
        padding: 50px 25px;
        margin: 30px auto;
        border: 1px solid #c3c3c3;
        text-align: center;
        width: 20%;
        display: inline-flex;
    }
    .e-file-select,
    .e-file-drop {
        display: none;
    }
    body .e-upload-drag-hover {
         outline: 2px dashed brown;
    }
    #uploadfile {
        width: 60%;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        display: inline-flex;
        margin-left: 5%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/getting-started-cs2" %}

Configure asynchronous settings

The uploader component process the files to upload in Asynchronous mode by default. Define the properties saveUrl and removeUrl to handle the save and remove action as follows.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" ></ejs-uploader>
  </div>
</template>
<script setup>
import { UploaderComponent as EjsUploader } from '@syncfusion/ej2-vue-
inputs';
const path =  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
        };
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" ></ejs-uploader>
  </div>
</template>
<script>
import { UploaderComponent } from '@syncfusion/ej2-vue-inputs';
export default {
    components: {
      'ejs-uploader': UploaderComponent
    },
    data: function() {
        return {
        path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        }
        }
    }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/getting-started-cs3" %}

Handle success and failed upload

You can handle the success and failure actions using the success and failure events. To handle these event, define the function and assign it to corresponding event as follows.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" :success= "onUploadSuccess" :failure=
"onUploadFailed" ></ejs-uploader>
  </div>
</template>
<script setup>
import { UploaderComponent as EjsUploader } from '@syncfusion/ej2-vue-
inputs';
const path =  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            };
const onUploadSuccess = (() => {
      console.log('Uploaded successfully');
});
const onUploadFailed = (() => {
      console.log('Upload fails');
});

</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" :success= "onUploadSuccess" :failure=
"onUploadFailed" ></ejs-uploader>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  </div>
</template>
<script>
import { UploaderComponent } from '@syncfusion/ej2-vue-inputs';
export default {
    components: {
      'ejs-uploader': UploaderComponent
    },
    data: function() {
        return {
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            }
        }
    },
      methods:{
        onUploadSuccess: function () {
            console.log('Uploaded successfully');
        },
        onUploadFailed: function () {
            console.log('Upload fails');
        }
      }
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/getting-started-cs4" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example that shows how to render the file upload and browse the files which you want to upload to the server.

See Also

- How to add additional data on upload
- Achieve file upload programmatically
- Achieve invisible upload

Getting Started with the Vue Uploader Component in Vue 3

This article provides a step-by-step guide for setting up a Vite project with a JavaScript environment and integrating the Syncfusion Vue Uploader component using the Composition API / Options API.

The Composition API is a new feature introduced in Vue.js 3 that provides an alternative way to organize and reuse component logic. It allows developers to write components as functions that use smaller, reusable functions called composition functions to manage their properties and behavior.

Copyright © 2001 -2024 Syncfusion Inc.

The Options API is the traditional way of writing Vue.js components, where the component logic is organized into a series of options that define the component's properties and behavior. These options include data, methods, computed properties, watchers, lifecycle hooks, and more.

Prerequisites

System requirements for Syncfusion Vue UI components

Set up the Vite project

A recommended approach for beginning with Vue is to scaffold a project using Vite. To create a new Vite project, use one of the commands that are specific to either NPM or Yarn.

`bash

npm create vite@latest

`

or

`bash

yarn create vite

`

Using one of the above commands will lead you to set up additional configurations for the project as below:

1.Define the project name: We can specify the name of the project directly. Let's specify the name of the project as my-project for this article.

`bash

? Project name: » my-project

`

2.Select Vue as the framework. It will create a Vue 3 project.

`bash

? Select a framework: » - Use arrow-keys. Return to submit.

Vanilla

Vue

React

Preact

Lit

Svelte

Others

`

3.Choose JavaScript as the framework variant to build this Vite project using JavaScript and Vue.

`bash

Copyright © 2001 -2024 Syncfusion Inc.

? Select a variant: » - Use arrow-keys. Return to submit.

JavaScript

TypeScript

Customize with create-vue ↗

Nuxt ↗

`

4.Upon completing the aforementioned steps to create the my-project, run the following command to install its dependencies:

`bash

cd my-project

npm install

`

or

`bash

cd my-project

yarn install

`

Now that my-project is ready to run with default settings, let's add Syncfusion components to the project.

Add Syncfusion Vue packages

Syncfusion Vue component packages are available at npmjs.com. To use Syncfusion Vue components in the project, install the corresponding npm package.

This article uses the Vue Uploader component as an example. To use the Vue Uploader component in the project, the @syncfusion/ej2-vue-inputs package needs to be installed using the following command:

`bash

npm install @syncfusion/ej2-vue-inputs --save

`

or

`bash

yarn add @syncfusion/ej2-vue-inputs

`

Import Syncfusion CSS styles

You can import themes for the Syncfusion Vue component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and Theme Studio. Refer to themes topic to know more about built-in themes and different ways to refer to themes in a Vue project.

Copyright © 2001 -2024 Syncfusion Inc.

In this article, Material theme is applied using CSS styles, which are available in installed packages. The necessary Material CSS styles for the Uploader component and its dependents were imported into the <style> section of **src/App.vue** file.

~/SRC/APP.VUE

```
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

The order of importing CSS styles should be in line with its dependency graph.

Add Syncfusion Vue component

Follow the below steps to add the Vue Uploader component using Composition API or Options API:

1.First, import and register the Uploader component in the script section of the **src/App.vue** file. If you are using the Composition API, you should add the setup attribute to the script tag to indicate that Vue will be using the Composition API.

COMPOSITION API (~/SRC/APP.VUE)

```
<script setup>
import { UploaderComponent as EjsUploader } from "@syncfusion/ej2-vue-
inputs";
</script>
```

OPTIONS API (~/SRC/APP.VUE)

```
<script>
import { UploaderComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: "App",
components: {
'ejs-uploader' : UploaderComponent,
}
}
</script>
```

2.In the template section, define the Uploader component with the dataSource property and column definitions.

~/SRC/APP.VUE

```
<template>
<div>
<div id="modalTarget" class="control-section; position:relative"
style="height:350px;">
<ejs-uploader ref="uploadObj" id='defaultfileupload'
name="UploadFiles"></ejs-uploader>
</div>
</div>
</template>
```

Copyright © 2001 -2024 Syncfusion Inc.

Here is the summarized code for the above steps in the **src/App.vue** file:

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div>
<div id="modalTarget" class="control-section; position:relative"
style="height:350px;">
<ejs-uploader ref="uploadObj" id='defaultfileupload'
name="UploadFiles"></ejs-uploader>
</div>
</div>
</template>
<script setup>
import { UploaderComponent as EjsUploader } from "@syncfusion/ej2-vue-
inputs";
const data = [];
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#app {
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 20%;
}
.control-section {
height: 100%;
min-height: 200px;
}
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div>
<div id="modalTarget" class="control-section; position:relative"
style="height:350px;">
<ejs-uploader ref="uploadObj" id='defaultfileupload'
name="UploadFiles"></ejs-uploader>
</div>
</div>
</template>
<script>
import { UploaderComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: "App",
components: {
'ejs-uploader': UploaderComponent,
},
```

Copyright © 2001 -2024 Syncfusion Inc.

```
data: function() {
return {  }
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#app {
color: #008cff;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 20%;
}
.control-section {
height: 100%;
min-height: 200px;
}
</style>
```

Run the project

To run the project, use the following command:

`bash

npm run dev

`

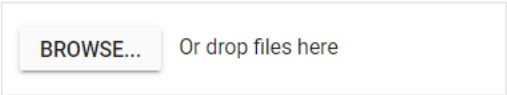
or

`bash

yarn run dev

`

The output will appear as follows:



Adding drop area

By default, the uploader component allows to upload files by drag the files from file explorer, and drop into the drop area. You can configure any other external element as drop target using dropArea property.

In the following sample, drop target is configured.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div>
<div id='droparea'>Drop files here to upload</div>
<ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:dropArea = "data[0].dropElement"></ejs-uploader>
</div>
</template>
<script setup>
import { UploaderComponent  as EjsUploader } from "@syncfusion/ej2-vue-
inputs";
const data = [{dropElement: '#dropArea'}]
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#container {
visibility: hidden;
padding-left: 5%;
width: 100%;
}
#loader {
color: #008cff;
font-family: 'Helvetica Neue','calibiri';
font-size: 14px;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.fileupload {
margin: 20px auto;
width: 400px;
}
#droparea {
padding: 50px 25px;
margin: 30px auto;
border: 1px solid #c3c3c3;
text-align: center;
width: 20%;
display: inline-flex;
}
.e-file-select,
.e-file-drop {
display: none;
}
body .e-upload-drag-hover {
outline: 2px dashed brown;
}
#uploadfile {
width: 60%;
display: inline-flex;
margin-left: 5%;
}
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

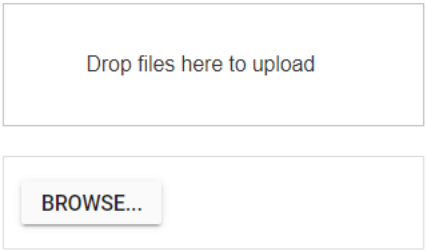
OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div>
<div id='droparea'>Drop files here to upload</div>
<ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:dropArea = "dropElement"></ejs-uploader>
</div>
</template>
<script>
import { UploaderComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: "App",
components: {
'ejs-uploader' : UploaderComponent,
},
data: function() {
return {
dropElement: '#droparea'
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
#container {
visibility: hidden;
padding-left: 5%;
width: 100%;
}
#loader {
color: #008cff;
font-family: 'Helvetica Neue','calibiri';
font-size: 14px;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.fileupload {
margin: 20px auto;
width: 400px;
}
#droparea {
padding: 50px 25px;
margin: 30px auto;
border: 1px solid #c3c3c3;
text-align: center;
width: 20%;
display: inline-flex;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.e-file-select,
.e-file-drop {
display: none;
}
body .e-upload-drag-hover {
outline: 2px dashed brown;
}
#uploadfile {
width: 60%;
display: inline-flex;
margin-left: 5%;
}
</style>
```

The output will appear as follows:



Configure asynchronous settings

The uploader component process the files to upload in Asynchronous mode by default. Define the properties saveUrl and removeUrl to handle the save and remove action as follows.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div>
<ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "data[0].path" ></ejs-uploader>
</div>
</template>
<script setup>
import { UploaderComponent as EjsUploader } from "@syncfusion/ej2-vue-
inputs";
const data = [{ path: { saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'}}]
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

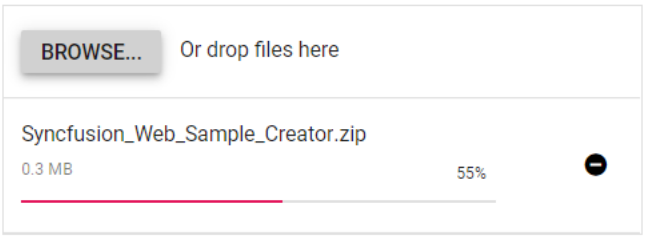
OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" ></ejs-uploader>
</div>
</template>
<script>
import { UploaderComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: "App",
components: {
'ejs-uploader': UploaderComponent,
},
data: function() {
return {
path: {
saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
}
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

The output will appear as follows:



Handle success and failed upload

You can handle the success and failure actions using the success and failure events. To handle these event, define the function and assign it to corresponding event as follows.

COMPOSITION API (~/SRC/APP.VUE)

```
<template>
<div>
<ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "data[0].path" :success= "onUploadSuccess" :failure=
"onUploadFailed" ></ejs-uploader>
</div>
</template>
<script setup>
import { UploaderComponent as EjsUploader} from "@syncfusion/ej2-vue-
inputs";
```

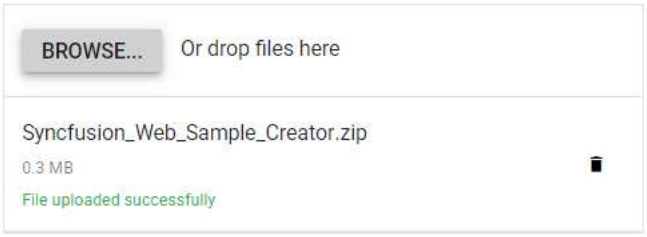
```
const data = [{ path: { saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'}}]
const onUploadSuccess = () => {
console.log('Uploaded successfully');
};
const onUploadFailed = () => {
console.log('Upload fails');
};
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

OPTIONS API (~/SRC/APP.VUE)

```
<template>
<div>
<ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" :success= "onUploadSuccess" :failure=
"onUploadFailed" ></ejs-uploader>
</div>
</template>
<script>
import { UploaderComponent } from "@syncfusion/ej2-vue-inputs";
//Component registeration
export default {
name: "App",
components: {
'ejs-uploader' : UploaderComponent,
},
data: function() {
return {
path:  {
saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
}
}
},
methods: {
onUploadSuccess: function () {
console.log('Uploaded successfully');
},
onUploadFailed: function () {
console.log('Upload fails');
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

The output will appear as follows:



See Also

- How to add additional data on upload
- Achieve file upload programmatically
- Achieve invisible upload

Async in Vue Uploader component

The uploader component allows you to upload the files asynchronously. The upload process requires save and remove action URL to manage the upload process in the server.

- The save action is necessary to handle the upload operation
- The remove action is optional, can handle the removed files from server

The File can be uploaded automatically or manually. For more information, you can refer to the Auto Upload section from the documentation.

Multiple file upload

By Default, the uploader component allows you to select and upload multiple files simultaneously. The selected files are organized in a list for every file selection until you clear it by clicking clear button that is shown in footer.

You can enable the multiple file selection by using `multiple` API.

The following example explains about multiple file upload settings.

In the following example, explains about multiple file upload settings.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" ></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
    data: function() {
        return {
```

```
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/asynchronous-cs1" %}

Single file upload

You can select and upload a single file by disabling the multiple file selection property. The file list item is removed for every selection and it always maintain a single file to upload.

The following example explains about single file upload settings.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" :multiple = 'multiple'></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
    data: function() {
        return {
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            },
            multiple: false
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/uploader/asynchronous-cs2" %}

Save Action

The save action handler upload the files that needs to be specified in the saveUrl property.

The save handler receives the submitted files and manages the save process in server. After uploading the files to server location, the color of the selected file name changes to green and the remove icon is changed as bin icon.

- When the file is uploaded successfully, the event “success” triggers to handle the operation after upload.
- When the file is failed to upload, the event “failure” triggers with information, which cause this failure.

You can cancel the upload process by setting the upload event argument **eventargs.cancel** to true.

```
<template>
<div>
<ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles" :asyncSettings= "path" ></ejs-uploader>
</div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
data: function() {
return {
path:  {
saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save'
}
}
}
}
</script>
```

Server-side configuration for save action

This section explains how to handle the server-side action for saving the file from server.

Copyright © 2001 -2024 Syncfusion Inc.

```c#
[AcceptVerbs("Post")]
public void Save()
{
try
{
if (HttpContext.Current.Request.Files.AllKeys.Length > 0)
{
var httpPostedFile = HttpContext.Current.Request.Files["UploadFiles"];
if (httpPostedFile != null)
{
var fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
var fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
if (!File.Exists(fileSavePath))
{
httpPostedFile.SaveAs(fileSavePath);
HttpResponse Response = HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusDescription = "File uploaded succesfully";
Response.End();
}
else
{
HttpResponse Response = HttpContext.Current.Response;
Response.Clear();
Response.Status = "400 File already exists";
Response.StatusCode = 400;
Response.StatusDescription = "File already exists";
Response.End();
}
}
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
}
catch (Exception e)
{
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusCode = 400;
Response.Status = "400 No Content";
Response.StatusDescription = e.Message;
Response.End();
}
}
```

Remove Action

The remove action is optional. Specify the URL to handle remove process from server. The remove handler receives the posted files and handle the remove operation in server.

- When the files are removed successfully from server, the success event triggers to denote the process has completed.
- When remove action fails, the event “failure” triggers with information, which cause failure in remove process.

You can differentiate the file operation whether the success event triggers from save or remove action in its arguments **eventArgs.operation**.

You can remove the files which is not uploaded locally by clicking the remove icon. In this case, the success or failure events will not be triggered.

```
<template>
<div>
<ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles" :asyncSettings= "path" ></ejs-uploader>
</div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
```

```
export default {
data: function() {
return {
path:  {
saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
}
}
}
}
</script>
`
```

Server-side configuration for remove action

This section explains how to handle the server-side action for removing the file from server.

```
`c#
[AcceptVerbs("Post")]
public void Remove()
{
try
{
var fileSave = "";
if (HttpContext.Current.Request.Form["cancel-uploading"] != null)
{
fileSave = HttpContext.Current.Server.MapPath("UploadingFiles");
}
else
{
fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
}
var fileName = HttpContext.Current.Request.Files["UploadFiles"].FileName;
var fileSavePath = Path.Combine(fileSave, fileName);
if (File.Exists(fileSavePath))
{
```

File.Delete(fileSavePath);

}

}

catch (Exception e)

{

HttpResponse Response = HttpContext.Current.Response;

Response.Clear();

Response.Status = "404 File not found";

Response.StatusCode = 404;

Response.StatusDescription = "File not found";

Response.End();

}

}

`

Auto Upload

By default, the uploader processes the files to upload once the files are selected and added in upload queue. To upload manually, disable the autoUpload property. When you disable this property, you can use the action buttons to call upload all or clear all actions manually. You can change those buttons text using the buttons property in the Uploader component.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" :autoUpload= 'autoUpload'></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
    data: function() {
        return {
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            },
            autoUpload: false
        }
    }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/asynchronous-cs3" %}

Sequential Upload

By default, the uploader component process multiple files to upload simultaneously. When you enable the sequentialUpload property, the selected files will process sequentially (one after the other) to the server. If the file uploaded successfully or failed, the next file will upload automatically in this sequential upload. This feature helps to reduce the upload traffic and reduce the failure of file upload.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" :sequentialUpload= 'sequentialUpload'></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
    data: function() {
        return {
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            },
            sequentialUpload: true
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/sequential-upload-cs1" %}

Preload Files

The uploader component allows you to preload the list of files that are uploaded in the server. The preloaded files are useful to view and remove the files from server that can be achieved by the files property. By default, the files are configured with uploaded successfully state on rendering file list. By default, the files are configured with uploaded successfully state on rendering file list. The following properties are mandatory to configure the preloaded files:

Copyright © 2001 -2024 Syncfusion Inc.

- Name
- Size
- Type

APP.VUE

```
<template>
  <div>
  <ejs-uploader id='preloadFiles' ref="uploadObj" name="UploadFiles"
:asyncSettings= "path" >
    <e-files>
        <e-uploadedfiles name='Nature' size=11000 type='.png'></e-
uploadedfiles>
        <e-uploadedfiles name='TypeScript Succintly' size=20000
type='.pdf'></e-uploadedfiles>
        <e-uploadedfiles name='ASP.NET Webhooks' size=35000
type='.docx'></e-uploadedfiles>
    </e-files>
    </ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
    data: function() {
        return {
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/asynchronous-cs4" %}

Adding additional HTTP headers with upload action

The Uploader component allows you to add the additional headers with save and remove action requests using the uploading and removing events, which helps to send validation token on file upload. Access the current request and set the request header within these events.

The following code block shows how to add the additional headers with save and remove action request.

`

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
<div>
<ejs-uploader id='preloadFiles' ref="uploadObj" name="UploadFiles" :asyncSettings= "path"
:uploading= "addHeaders" :removing= "addHeaders" ></ejs-uploader>
</div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
data: function() {
return {
path:  {
saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
}
}
},
methods: {
addHeaders: function (args) {
args.currentRequest.setRequestHeader('custom-header', 'Syncfusion');
}
}
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
`
```

Copyright © 2001 -2024 Syncfusion Inc.

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

See Also

- How to add additional data on upload
- How to add confirm dialog to remove the files
- Check the MIME type of file before uploading it
- How to open and edit the uploaded files

Chunk upload in Vue Uploader component

The Uploader sends the large file split into small chunks and transmits to the server using AJAX. You can also pause, resume, and retry the failed chunk file.

* The chunk upload works in asynchronous upload only.

- This feature is available from the Essential Studio Vol 2, 2018 release.

To enable the chunk upload, set the size to chunkSize option of the upload and it receives the value in bytes.

APP.VUE

```
<template>
    <div class="col-lg-8 control-section uploader chunk">
        <div class="control_wrapper">
            <ejs-uploader id='chunkupload' name="UploadFiles"
:asyncSettings= "path" ref="uploadObj">         </ejs-uploader>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
 data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove',
            chunkSize: 500000
          }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
.chunk .control_wrapper {
    max-width: 450px;
    min-width: 245px;
    margin: auto;
}
#chunkupload .e-upload.e-control {
    position: relative;
    margin: 15px 0;
}
.control-section .uploader.col-lg-8 {
    padding: 20px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/chunk-upload-cs1" %}

The chunk upload functionality separates the selected files into blobs of the data or chunks. These chunks are transmitted to the server using an AJAX request.

The chunks are sent in **sequential** order, and the next chunk can be sent to the server according to the success of the previous chunk. If any one of the chunk failed, then the remaining chunk cannot be sent to the server. The chunkSuccess or chunkFailure event will be triggered when the chunk is sent to the server successfully or failed. If all the chunks are sent to the server successfully, the uploader success event is triggered.

Chunk upload will work when the selected file size is greater than the specified chunk size. otherwise, it upload the files normally.

Additional configurations

To modify the chunk upload, the following options can be used.

- **RetryAfterDelay** - If error occurs while sending any chunk request from JavaScript, hold the operation for 500 milliseconds (by default), and retry the operation using chunk. This can be achieved by using the asyncSettings.retryAfterDelay property. You can modify the holding time interval in milliseconds.
- **RetryCount** - Specifies the number of retry actions performed when the file fails to upload. By default, retry action is performed 3 times. If the file fails to upload continuously, the request is aborted and the uploader failure event will trigger.

Copyright © 2001 -2024 Syncfusion Inc.

The following sample specifies the chunk upload delay with 3000 milliseconds and the retry count is 5. The failure event is triggered as the wrong saveUrl is used.

APP.VUE

```
<template>
    <div class="col-lg-8 control-section uploader chunk">
        <div class="control_wrapper">
            <ejs-uploader id='chunkupload' name="UploadFiles"
:asyncSettings= "path" ref="uploadObj">
            </ejs-uploader>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
 data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove',
            chunkSize: 500000,
             // set count for automatic retry when chunk upload failed
            retryCount: 5,
            // set time delay for automatic retry when chunk upload failed
            retryAfterDelay: 3000
          }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
.chunk .control_wrapper {
    max-width: 450px;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    min-width: 245px;
    margin: auto;
}
#chunkupload .e-upload.e-control {
    position: relative;
    margin: 15px 0;
}
.control-section .uploader.col-lg-8 {
    padding: 20px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/chunk-upload-cs2" %}

Resumable upload

Allows you to resume an upload operation after a network failure or manually interrupts (pause) the upload. You can perform pause and resume upload actions using public methods (pause and resume) and UI interaction. The pause icon is enabled after the upload begins.

This pause and resume features available only when the chunk upload is enabled.

APP.VUE

```
<template>
    <div class="col-lg-8 control-section uploader chunk">
        <div class="control_wrapper">
            <ejs-uploader id='chunkupload' name="UploadFiles"
:asyncSettings= "path" ref="uploadObj"></ejs-uploader>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
 data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove',
            chunkSize: 102400
          }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/chunk-upload-cs3" %}

Cancel upload

The uploader component allows you to cancel the uploading file. This can be achieved by clicking the cancel icon or using the cancel method. The cancelling event will be fired whenever the file upload request is canceled. While canceling the upload request, the partially uploaded file is removed from the server.

When the request fails, the pause icon is changed to retry icon. By clicking the retry icon, sends the failed chunk request again to the server and upload started from where it is failed. You can retry the canceled upload request again using retry UI or retry methods. But, if you retry this, the file upload action again starts from initial.

The following example explains about chunk upload with cancel support.

APP.VUE

```
<template>
    <div class="col-lg-8 control-section uploader chunk">
        <div class="control_wrapper">
            <ejs-uploader id='chunkupload' name="UploadFiles"
:asyncSettings= "path" ref="uploadObj"></ejs-uploader>
        </div>
    </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
 data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove',
            chunkSize: 102400
          }
        }
    }
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/chunk-upload-cs4" %}

The retry action has different working behavior for chunk upload and default upload.

- Chunk upload - Retries to upload the failed request where it is failed previously.
- Default upload - Retries to upload the failed file again from initial.

Server-Side configurations

The server-side implementation entirely depends on the application requirements and logic. The following code snippet provides the server-side logic to handle the chunk upload using the uploader components.

`c#

// Server configuration for upload a file.

public void Save()

{

try

{

if (HttpContext.Current.Request.Files.AllKeys.Length > 0)

{

var httpPostedChunkFile = HttpContext.Current.Request.Files["chunkFile"];

if (httpPostedChunkFile != null)

{

var saveFile = HttpContext.Current.Server.MapPath("UploadingFiles");

Copyright © 2001 -2024 Syncfusion Inc.

```
// Save the chunk file in temporery location with .part extension
var SaveFilePath = Path.Combine(saveFile, httpPostedChunkFile.FileName + ".part");
var chunkIndex = HttpContext.Current.Request.Form["chunk-index"];
if (chunkIndex == "0")
{
httpPostedChunkFile.SaveAs(SaveFilePath);
}
else
{
// Merge the current chunk file with previous uploaded chunk files
MergeChunkFile(SaveFilePath, httpPostedChunkFile.InputStream);
var totalChunk = HttpContext.Current.Request.Form["total-chunk"];
if (Convert.ToInt32(chunkIndex) == (Convert.ToInt32(totalChunk) - 1))
{
var savedFile = HttpContext.Current.Server.MapPath("UploadedFiles");
var originalFilePath = Path.Combine(savedFile, httpPostedChunkFile.FileName);
// After all the chunk files completely uploaded, remove the .part extension and move this file into save location
System.IO.File.Move(SaveFilePath, originalFilePath);
}
}
HttpResponse ChunkResponse = HttpContext.Current.Response;
ChunkResponse.Clear();
ChunkResponse.ContentType = "application/json; charset=utf-8";
ChunkResponse.StatusDescription = "File uploaded succesfully";
ChunkResponse.End();
}
var httpPostedFile = HttpContext.Current.Request.Files["UploadFiles"];
if (httpPostedFile != null)
{
var fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
var fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
if (!File.Exists(fileSavePath))
```

```
{
httpPostedFile.SaveAs(fileSavePath);
HttpResponse Response = HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusDescription = "File uploaded succesfully";
Response.End();
}
else
{
HttpResponse Response = HttpContext.Current.Response;
Response.Clear();
Response.Status = "400 File already exists";
Response.StatusCode = 400;
Response.StatusDescription = "File already exists";
Response.End();
}
}
}
}
catch (Exception e)
{
HttpResponse Response = HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusCode = 400;
Response.Status = "400 No Content";
Response.StatusDescription = e.Message;
Response.End();
}
}
// Server configuration for remove a uploaded file
public void Remove()
```

```
{
try
{
var fileSave = "";
if (HttpContext.Current.Request.Form["cancel-uploading"] != null)
{
fileSave = HttpContext.Current.Server.MapPath("UploadingFiles");
} else
{
fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
}
var fileName = HttpContext.Current.Request.Files["UploadFiles"].FileName;
var fileSavePath = Path.Combine(fileSave, fileName);
if (File.Exists(fileSavePath))
{
File.Delete(fileSavePath);
}
}
catch (Exception e)
{
HttpResponse Response = HttpContext.Current.Response;
Response.Clear();
Response.Status = "404 File not found";
Response.StatusCode = 404;
Response.StatusDescription = "File not found";
Response.End();
}
}
// Merge the current chunk file with previous uploaded chunk files
public void MergeChunkFile(string fullPath, Stream chunkContent)
{
try
{
```

```
using (FileStream stream = new FileStream(fullPath, FileMode.Append, FileAccess.Write,
FileShare.ReadWrite))
{
using (chunkContent)
{
chunkContent.CopyTo(stream);
}
}
}
catch (IOException ex)
{
throw ex;
}
}
`
```

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

File source in Vue Uploader component

Paste to upload

The uploader component allows you to upload the files using the select or drop files option from the file explorer. It also supports pasting to upload the image files. You can upload any currently copied images in the clipboard.

When you paste the image, it will be saved in the server with the filename as image.png. The file name can

be renamed in the server end. You can generate a random name for the file name using getUniqueID method.

Refer to the following example.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" :autoUpload= 'autoUpload'
:uploading="onUploadBegin"></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin, UploadingEventArgs } from '@syncfusion/ej2-vue-
inputs';
```

```
import { getUniqueID } from '@syncfusion/ej2-base';
Vue.use(UploaderPlugin);
export default {
    data: function() {
        return {
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            },
            autoUpload: false
        }
    },
    methods:{
        onUploadBegin: function(args) {
            // check whether the file is uploading from paste.
            if (args.fileData.fileSource === 'paste') {
                let newName: string =
getUniqueID(args.fileData.name.substring(0,
args.fileData.name.lastIndexOf('.'))) + '.png';
                args.customFormData = [{'fileName': newName}];
            }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/asynchronous-cs5" %}

Save action for paste to upload

`c#

public void Save()

{

var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];

var fileSave = System.Web.HttpContext.Current.Server.MapPath("UploadedFiles");

var fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);

if (!System.IO.File.Exists(fileSavePath))

{

httpPostedFile.SaveAs(fileSavePath);

var newName = System.Web.HttpContext.Current.Request.Form["fileName"];

var filePath = Path.Combine(fileSavePath.Substring(0, fileSavePath.LastIndexOf("//")), newName);

// Rename the file

Copyright © 2001 -2024 Syncfusion Inc.

```
System.IO.File.Move(fileSavePath, newName);

HttpResponse Response = System.Web.HttpContext.Current.Response;

Response.Clear();

Response.ContentType = "application/json; charset=utf-8";

Response.StatusDescription = fileSavePath;

Response.End();

}

}
`
```

Directory upload

The uploader component allows you to upload all files in the folders to server by using the directoryUpload property. When this property is enabled, the uploader component processes the files by iterating through the files and sub-directories in a directory. It allows you to select only folders instead of files to upload.

The directory upload is available only in browsers that supports **HTML5 directory**. The uploader will process directory upload by dragging and dropping in the Edge browser. Refer to the following example to upload files to the server.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" :directoryUpload = 'directory'></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
    data: function() {
        return {
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            },
            directory: true
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

Copyright © 2001 -2024 Syncfusion Inc.

{% previewsample "page.domainurl/code-snippet/uploader/asynchronous-cs6" %}

Save action for directory upload

```c#
public void Save() {
var httpPostedFile = HttpContext.Current.Request.Files["UploadFiles"];
var fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
// split the folders by using file name
string[] folders = httpPostedFile.FileName.Split('/');
string fileSavePath = "";
if (folders.Length > 1)
{
for (var i = 0; i < folders.Length - 1; i++)
{
var newFolder = Path.Combine(fileSave, folders[i]);
// create folder
Directory.CreateDirectory(newFolder);
fileSave = newFolder;
}
fileSavePath = Path.Combine(fileSave, folders[folders.Length - 1]);
}
else
{
fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
}
if (!System.IO.File.Exists(fileSavePath))
{
// save file in the corresponding server location
httpPostedFile.SaveAs(fileSavePath);
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
// Sending the file path to client side
```

Copyright © 2001 -2024 Syncfusion Inc.

Response.StatusDescription = fileSavePath;

Response.End();

}

}

`

Drag and drop

The uploader component allows you to drag and drop the files to upload. You can drag the files from file explorer and drop into the drop area. By default, the uploader component act as drop area element. The drop area gets highlighted when you drag the files over drop area.

Custom drop area

The uploader component allows you to set external target element as drop area using the dropArea property. The element can be represented as HTML element or element's id.

APP.VUE

```
<template>
  <div>
    <div id='droparea'  >
            Drop files here to upload
    </div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" :dropArea= 'dropTarget' ></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
    data: function() {
        return {
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            },
            dropTarget: '#droparea'
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
```

```
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .fileupload {
        margin: 20px auto;
        width: 400px;
    }
    #droparea {
        padding: 50px 25px;
        margin: 30px auto;
        border: 1px solid #c3c3c3;
        text-align: center;
        width: 20%;
        display: inline-flex;
    }
    .e-file-select,
    .e-file-drop {
        display: none;
    }
    body .e-upload-drag-hover {
         outline: 2px dashed brown;
    }
    #uploadfile {
        width: 60%;
        display: inline-flex;
        margin-left: 5%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/draganddrop-cs1" %}

Customize drop area

You can customize the appearance of drop area by overriding the default drop area styles. The class “” and “” is available to handle this customization.

APP.VUE

```
<template>
  <div id="UploaderDropTarget">
         <div id='dropArea' style='height: auto; overflow: auto'>
            <span id='drop'> Drop files here or <a href=""
id='browse'><u>Browse</u></a> </span>
        <ejs-uploader id='templateupload' name="UploadFiles" :asyncSettings=
"path" ref="uploadObj" :dropArea= "dropElement">
        </ejs-uploader>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
    data: function() {
        return {
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            },
            dropElement: '#drop'
        }
    },
     mounted: function () {
        document.getElementById('browse').onclick = () => {
        document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('#templateupload').click();
        return false;
    }
}
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
   #UploaderDropTarget {
        min-height: 50px;
        padding-top: 15px;
        position: relative;
        display: inline-block;
        font-size: 13px;
        top: -3px;
        width: 100%;
    }
.e-file-select-wrap {
    display: none;
}
#dropArea .e-upload {
    border: 0;
    margin-top: 15px;
}
#drop {
    padding-left: 30%;
}
#dropArea {
    min-height: 18px;
    border: 1px dashed #c3c3cc;
    padding-top: 15px;
    margin: 20px auto;
    width: 400px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/custom-drop-area-cs1" %}

Copyright © 2001 -2024 Syncfusion Inc.

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

See Also

- Achieve file upload programmatically
- Validate image/* on drop

Validation in Vue Uploader component

The uploader component validate the selected files extension and size using the allowedExtensions, minFileSize and maxFileSize properties. The files can be validated before uploading to the server and can be ignored on uploading. Also, you can validate the files by setting the HTML attributes to the original input element. The validation process occurs on drag-and-drop the files also.

File type

You can allow the specific types of files alone to upload using the allowedExtensions property. The extension can be represented as collection by comma separators. The uploader component filters the selected or dropped files matched against the specified file types and processes the upload operation. The validation happens when you specify value to inline attribute accept to original input element.

APP.VUE

```
<template>
    <div class="col-lg-8 control-section uploader chunk">
        <div class="control_wrapper">
            <ejs-uploader id='validation' name="UploadFiles" :asyncSettings=
"path" ref="uploadObj" :allowedExtensions = "extensions"></ejs-uploader>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
import { FileInfo } from '@syncfusion/ej2-vue-inputs/uploader';
Vue.use(UploaderPlugin);
export default {
 data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
          extensions: '.doc, .docx, .xls, .xlsx'
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/file-type-cs1" %}

File size

The uploader component allows you to validate the files based on its size. The validation helps to restrict uploading large files or empty files to the server. The size can be represented in bytes. By default, the uploader component allows you to upload **minimum file size** as 0 byte and **maximum file size** as 28.4 MB using using minFileSize and maxFileSize properties.

APP.VUE

```
<template>
    <div class="col-lg-8 control-section uploader chunk">
        <div class="control_wrapper">
            <ejs-uploader id='validation' name="UploadFiles" :asyncSettings=
"path" ref="uploadObj"  minFileSize=10000 maxFileSize=4000000>
            </ejs-uploader>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
import { FileInfo } from '@syncfusion/ej2-vue-inputs/uploader';
Vue.use(UploaderPlugin);
export default {
 data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/file-size-cs1" %}

Maximum files count

You can restrict uploading the maximum number of files using the **selected** event. In the selected event arguments, you can get the currently selected files details using the getFilesData(). You can modify the files details and assign the modified file list to the eventArgs.modifiedFilesData.

APP.VUE

```
<template>
    <div class="col-lg-8 control-section uploader chunk">
        <div class="control_wrapper">
            <ejs-uploader id='validation' name="UploadFiles" :autoUpload=
"autoUpload" :asyncSettings= "path" ref="uploadObj" :selected=
"onFileSelect" :removing= "onFileRemove">
            </ejs-uploader>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
import { FileInfo } from '@syncfusion/ej2-vue-inputs/uploader';
Vue.use(UploaderPlugin);
export default {
  data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
          autoUpload: false
        }
    },
    methods: {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        onFileSelect: function(args) {
            args.filesData.splice(5);
            let filesData = this.$refs.uploadObj.getFilesData();
            let allFiles = filesData.concat(args.filesData);
            if (allFiles.length > 5) {
                for (let i = 0; i < allFiles.length; i++) {
                    if (allFiles.length > 5) {
                        allFiles.shift();
                    }
                }
                args.filesData = allFiles;
                args.modifiedFilesData = args.filesData;
            }
            args.isModified = true;
        },
        onFileRemove: function (args) {
            args.postRawFile = false;
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/file-count-cs1" %}

Duplicate files

You can validate the duplicate files before uploading to server using the selected event. Compare the selected files with the existing files data and filter the file list by removing the duplicate files.

APP.VUE

```
<template>
    <div class="col-lg-8 control-section uploader chunk">
        <div class="control_wrapper">
            <ejs-uploader id='validation' name="UploadFiles" :autoUpload=
"autoUpload" :asyncSettings= "path" ref="uploadObj"
            :selected= "onFileSelect">
```

```
            </ejs-uploader>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
import { FileInfo } from '@syncfusion/ej2-vue-inputs/uploader';
import { detach, isNullOrUndefined, enableRipple } from '@syncfusion/ej2-
base';
enableRipple(true);
Vue.use(UploaderPlugin);
export default {
  data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
          autoUpload: false
        }
    },
    methods: {
        onFileSelect: function(args) {
        let existingFiles: FileInfo[] = this.$refs.uploadObj.getFilesData();
    for (let i: number = 0; i < args.filesData.length; i++) {
        for(let j: number = 0; j < existingFiles.length; j++) {
            if (!isNullOrUndefined(args.filesData[i])) {
                if (existingFiles[j].name == args.filesData[i].name) {
                    args.filesData.splice(i, 1);
                }
            }
        }
    }
    existingFiles = existingFiles.concat(args.filesData);
    args.modifiedFilesData = existingFiles;
    args.isModified = true;
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
```

```
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/validation-cs1" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

See Also

- Validate image/* on drop
- Determine whether uploader has file input (required validation)
- Check file size before uploading it
- Check the MIME type of file before uploading it

Form support in Vue Uploader component

The uploader component works with HTML form like default file input.

The following configuration is must to make the uploader work inside the form.

- saveUrl and removeUrl must be null.
- autoUpload must be disabled.
- name attribute must be added in input element.

The selected or dropped files are received as a collection in form action when the form is submitted. The form action handles the server-side operations that manage the file upload process. When you reset the form, the file list and data will be cleared.

APP.VUE

```
<template>
  <div >
  <div class="col-lg-12 control-section form-support">
        <div class="control_wrapper" id="control_wrapper">
            <!-- Initialize Uploader -->
            <form id="form1" action="saveFiles.ashx" method="post">
                    <h4 class="form-title">Photo Contest</h4>
                <div class="form-group" style="padding-top: 11px;">
                        <div class="e-float-input">
                            <input type="text" id="name" name="Name" data-
required-message="* Enter your name" required="" data-msg-
containerid="nameError">
                            <span class="e-float-line"></span>
                            <label class="e-float-text e-label-top"
for="name">Name</label>
                        </div>
                        <div id="nameError"></div>
                    </div>
```

```
                    <div class="form-group" style="padding-top: 11px;">
                            <div class="e-float-input">
                                <input type="email" id="email" name="Email"
data-validation="email" data-required-message="* Enter your email"
required="" data-msg-containerid="mailError">
                                <span class="e-float-line"></span>
                                <label class="e-float-text e-label-top"
for="email">Email</label>
                            </div>
                            <div id="mailError"></div>
                    </div>
                    <div class="form-group" style="padding-top: 11px;">
                            <div class="e-float-input">
                                <input type="text" id="mobileno"
name="MobileNo" data-required-message="* Enter your mobile number"
required="" data-msg-containerid="noError">
                                <span class="e-float-line"></span>
                                <label class="e-float-text e-label-top"
for="mobileno">Mobile No</label>
                            </div>
                            <div id="noError"></div>
                    </div>
                    <div class="form-group" style="padding-top: 11px;">
                            <div class="e-float-input upload-area">
                                <input type="text" id="upload" name="upload"
readonly="" data-required-message="* Select any file" required="" data-msg-
containerid="uploadError">
                                <ejs-uploader id='fileupload'
name="UploadFiles" :autoUpload= "isAuto" :selected= "onFileSelect"
                                 allowedExtensions= "image/*" :dropArea =
"dropElement" :multiple= 'false' ></ejs-uploader>
                                <button id="browse" class="e-control e-btn
e-info">Browse...</button>
                                <span class="e-float-line"></span>
                                <label class="e-float-text e-label-top"
for="upload">Choose a file</label>
                            </div>
                            <div id="uploadError"></div>
                    </div>
                    <div class="form-group" style="padding-top: 11px;">
                            <div class="e-float-input">
                                <textarea class="address-field" rows="4"
id="Address" name="Address"></textarea>
                                <span class="e-float-line"></span>
                                <label class="e-float-text e-label-top"
for="address">Address</label>
                            </div>
                    </div>
                </form>
                <div class="submitBtn">
                    <button class="submit-btn e-btn" id="submit-
btn">Submit</button>
                    <div class="desc"><span>*This button is not a submit
type and the form submit handled from externally.</span></div>
                </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
                <ejs-dialog id='uploadAlert' :header='header'
showCloseIcon=true :width= 'width' :content= 'dlgContent' :target=
'dlgTarget'
                :isModal= 'Modal' :visible="false" :animationSettings=
'animation' ref="dialogObj" >
                </ejs-dialog>
        </div>
    </div>
</div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
import { FormValidator  } from '@syncfusion/ej2-inputs';
import { DialogPlugin } from '@syncfusion/ej2-vue-popups';
Vue.use(UploaderPlugin);
Vue.use(DialogPlugin);
export default {
    data: function() {
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
          dropElement: '.control-fluid',
          extensions: '.jpg, .png',
          isAuto: false,
          formObj: '',
          header: 'Success',
          animation: {effect: 'zoom'},
          Modal: true,
          dlgTarget: '.control_wrapper',
          dlgContent: 'Your details have been updated successfully, Thank
you.',
          width: '335px',
          options : {
        //Initialize the CustomPlacement.
            customPlacement: function(inputElement, errorElement) {
                inputElement = inputElement.closest('.form-
group').querySelector('.error');
                inputElement.parentElement.appendChild(errorElement);
            },
                rules: {
                    'Name': {
                        required: true
                    },
                    'Email': {
                        required: true
                    },
                    'upload': {
                        required: true
                    }
                }
            }
        }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    },
    mounted: function () {
        document.getElementById('browse').onclick = () => {
            document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
            return false;
        };
        let localObj = this;
        document.getElementById('submit-btn').onclick = function() {
            localObj.onFormSubmit();
        };
        this.formObj = new FormValidator('#form1', this.options);
    },
    methods:{
        onFormSubmit: function() {
            let formStatus = this.formObj.validate();
            if (formStatus) {
                this.formObj.element.reset();
                this.$refs.dialogObj.show();
            }
        },
        onFileSelect: function(args) {
            let inputElement = document.getElementById('upload');
            inputElement.value = args.filesData[0].name;
        },
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .form-support .control_wrapper {
            max-width: 400px;
            margin: auto;
        }
        .form-support .address-field {
            max-height: 50px;
            resize: none;
        }
        .form-support #control_wrapper {
                max-width: 500px;
```

```
            margin: auto;
            border: 0.5px solid #BEBEBE;
            box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.36);
            padding: 1% 4% 2%;
            background: #f9f9f9;
        }
        .highcontrast .form-support #control_wrapper {
            background: #000000;
        }
        .form-support .upload-area, .e-bigger .form-support .upload-area {
            width: 73%;
        }
        .form-support .e-error {
            padding-top:3px;
        }
        .form-support .e-upload {
            width: 100%;
            position: relative;
            margin-top: 15px;
        }
        .form-support .control_wrapper .e-upload .e-upload-drag-hover {
            margin: 0;
        }
        .form-support .submit-btn {
            margin-top: 15px;
        }
        .form-support .submitBtn .desc {
            margin: 2% 20% 0% 18%;
        }
        @media only screen and (max-width: 500px) {
            .form-support .submitBtn .desc {
                margin: 12px;
            }
            .form-support .upload-area, .e-bigger .upload-area {
                width: 60%;
            }
        }
        .form-support .submitBtn {
            position: relative;
            text-align: center;
        }
        .form-support .success .successmsg {
            border: 0.5px solid green;
            padding: 10%;
            color: green;
        }
        .form-support form#form1 {
            position: relative;
            top: 14%;
        }
        .form-support .e-upload {
            display: none;
        }
        .form-support input.choose-file{
            width: 60%;
        }
        .form-support button#browse {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            float: right;
            margin-right: -115px;
            margin-top: -29px;
            position: relative;
        }
        .e-bigger.bootstrap .form-support button#browse {
            top: -12px;
        }
        .e-bigger.material .form-support  button#browse {
            top: -8px;
        }
        .e-bigger.fabric .form-support button#browse, .e-bigger.highcontrast
.form-support button#browse {
            top: -11px;
            left: 2%;
        }
        .fabric .form-support #browse, .highcontrast .form-support #browse {
            top: -3px;
        }
        .bootstrap .form-support #browse {
            top: -6px;
        }
        .form-support .form-title {
            text-align: center;
        }
        #form1 .e-float-input:not(.e-input-group), #form1 .e-float-input.e-
control-wrapper:not(.e-input-group) {
            display: inherit;
        }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/formsupport-cs1" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Template in Vue Uploader component

You can customize the default appearance of the uploader using a template along with buttons.

File list template

The template property is used to customize the default appearance of each file in the list. It can be represented as the HTML element or string. The selected or dropped files are displayed as per the template layout provided. The remove and progress bar action is handled using the corresponding events when the template is defined.

For example, you can display file type icon along with default UI elements.

APP.VUE

```
<template>
    <div>
        <div class="col-lg-12 control-section uploader customdroparea">
            <div class="control_wrapper">
                <div class="sample_wrapper">
                    <div id="dropArea">
```

```
                        <span id="drop" class="droparea"> Drop files here or
<a href="" id="browse"><u>Browse</u></a> </span>
                        <ejs-uploader id='templateupload' name="UploadFiles"
:template='fileTemplate' :allowedExtensions= 'extensions' :asyncSettings=
"path" ref="uploadObj" :dropArea= "dropArea" :success= "onSuccess" :failure=
"onFailure" :progress = "onProgress" :removing= "onFileRemove">
                        </ejs-uploader>
                    </div>
                </div>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { UploaderPlugin, SelectedEventArgs } from '@syncfusion/ej2-vue-
inputs';
import { createElement, detach, EventHandler } from '@syncfusion/ej2-base';
import { FileInfo } from '@syncfusion/ej2-vue-inputs/uploader';
Vue.use(UploaderPlugin);
var demoVue = Vue.component("demo", {
    template: `<div class='container'>
                    <span class='wrapper'>
                    <span :class="['icon sf-icon-' +data.type]"></span>
                    <span class='name file-name'>{{data.name}} (
{{data.size}} bytes)</span>
                    <span class='upload-status'>{{data.status}}</span>
                    <span class='e-icons e-file-remove-btn'
title='Remove'></span>
                    </span>
                </div>`,
  data() {
    return {
      data: {}
    };
  }
});
export default {
    data: function() {
        return {
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            },
            extensions: '.pdf, .png, .txt',
            dropArea: "#dropArea",
            fileTemplate: function (){
                return { template: demoVue }
            }
        }
    },
    mounted: function () {
        document.getElementById('browse').onclick = function() {
            document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
```

```
            return false;
        };
        document.getElementById('dropArea').onclick = (args) => {
            if (args.target.classList.contains('e-file-delete-btn')) {
                for (var i = 0; i <
this.$refs.uploadObj.getFilesData().length; i++) {
                    if (args.target.closest('li').getAttribute('data-file-
name') === this.$refs.uploadObj.getFilesData()[i].name) {

this.$refs.uploadObj.remove(this.$refs.uploadObj.getFilesData()[i]);
                    }
                }
            }
            else if (args.target.classList.contains('e-file-remove-btn')) {
                detach(args.target.closest('li'));
            }
        };
    },
    methods:{
        onSuccess: function(args) {
            let li = this.getLiElement(args);
            li.querySelector('.upload-status').innerHTML = args.file.status;
            li.querySelector('.upload-status').classList.add('upload-
success');
        },
        onFailure: function(args) {
            let li = this.getLiElement(args);
            li.querySelector('.upload-status').innerHTML = args.file.status;
            li.querySelector('.upload-status').classList.add('upload-
failed');
        },
        getLiElement: function(args) {
            let liElements = document.getElementsByClassName('e-
upload')[0].querySelectorAll('.e-upload-files > li');
            let li;
            for (let i= 0; i < liElements.length; i++) {
                if ( liElements[i].getAttribute('data-file-name') ===
args.file.name ) {
                li = liElements[i];
                }
            }
            return li;
        },
        onProgress: function(args) {
            let progressValue = Math.round((args.e.loaded / args.e.total) *
100) + '%';
            let li = this.getLiElement(args);
            li.querySelector('.upload-status').innerHTML = args.file.status
+ '(' + progressValue + ' )';
        },
        onFileRemove: function (args) {
            args.postRawFile = false;
        }
    }
}
</script>
<style>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
.sf-icon-bmp:before { content: "\e700"; }
.sf-icon-xlsx:before, .sf-icon-XLSX:before { content: "\e701"; }
.sf-icon-avi:before, .sf-icon-AVI:before { content: "\e702"; }
.sf-icon-doc:before, .sf-icon-DOC:before { content: "\e703"; }
.sf-icon-exe:before { content: "\e704"; }
.sf-icon-mp4:before { content: "\e705"; }
.sf-icon-zip:before, .sf-icon-ZIP:before { content: "\e706"; }
.sf-icon-tar:before { content: "\e707"; }
.sf-icon-docx:before { content: "\e708"; }
.sf-icon-jpg:before { content: "\e709"; }
.sf-icon-png:before { content: "\e70a"; }
.sf-icon-gif:before { content: "\e70b"; }
.sf-icon-pdf:before { content: "\e70c"; }
.sf-icon-txt:before { content: "\e70d"; }
.sf-icon-jpeg:before { content: "\e70e"; }
.sf-icon-xls:before { content: "\e70f"; }
.sf-icon-mp3:before { content: "\e710"; }
 .customdroparea .e-upload {
      border-top: none;
      border-bottom: none;
    }
    .customdroparea .e-upload .e-upload-files {
        border-top: none;
    }
    .customdroparea {
        margin: auto 160px;
    }
    .customdroparea #dropArea .e-upload .e-upload-files .e-upload-file-list
{
        min-height: 0;
    }
    .customdroparea #drop {
        padding: 6% 27% 6% 22%;
        display: block;
        border: 1px dashed #c3c3cc;
    }
    .customdroparea #drop a {
        color: blue;
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    .highcontrast .customdroparea #drop a {
        color:#ffd939;
    }
    .customdroparea #UploaderDropTarget {
        min-height: 50px;
        padding-top: 15px;
        position: relative;
        display: inline-block;
        font-size: 13px;
        top: -29px;
        width: 400px;
    }
    .customdroparea .e-file-select-wrap {
        display: none;
    }
    .customdroparea span.dropText {
        position: relative;
        left: 31%;
        top: 101px;
    }
    .customdroparea .dropArea_wrap {
        width: 400px;
        height:250px;
        background-color: lightgrey;
        display: inline-block;
        margin: 22px;
        border: 1px dashed grey;
    }
    .highcontrast  .customdroparea .dropArea_wrap {
        background-color: #000;
    }
   .customdroparea .font-icons {
        position: relative;
        top: 25%;
        left: 32%;
        width: 150px;
   }
    .customdroparea .upload-failed {
        color: #d9534f;
    }
    .customdroparea .e-upload .wrapper .upload-success {
        color: #107c10;
    }
    .customdroparea span.upload-status {
        left: 45px;
        position: relative;
        font-size: 12px;
        display: block;
        padding: 5px;
        top: -8px;
    }
    #UploaderDropTarget .e-upload .e-upload-files .e-file-reload-btn:hover,
    #dropArea .e-upload .e-upload-files .e-file-reload-btn {
        display: none;
    }
    #UploaderDropTarget .e-file-abort-btn.e-upload-progress {
        background-image: none;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        cursor: default;
        opacity: .35;
    }
    #UploaderDropTarget .e-icons.e-file-delete-btn:hover {
        background: none;
    }
    .customdroparea .e-upload .e-upload-files .e-file-remove-btn.e-icons,
.customdroparea .e-upload .e-upload-files .e-file-delete-btn.e-icons {
        top: 50%;
    }
    @font-face {
        font-family: 'FiletypeV2 Font';
        src:
        url(data:application/x-font-ttf;charset=utf-
8base64,AAEAAAAKAIAAAwAgT1MvMj1tSgIAAAEoAAAAVmNtYXDnYOfNAAAByAAAAFZnbHlmfTt
llAAAAkgAACpgaGVhZBBa3XUAAADQAAAANmhoZWEH1AQTAAAArAAAACRobXR4SAAAAAAAAYAAAAB
IbG9jYWOGV2wAAAIgAAAAJm1heHABNQGCAAABCAAAACBuYW1l8fKKOQAALKgAAAKFcG9zdGQHrBs
AAC8wAAAAkAABAAAEAAAAAFwEAAAAAADdwABAAAAAAAAAAAAAAAAAAAAEgABAAAAAQAAK7wGlF8
PPPUACwQAAAAAANbizEIAAAAA1uLMQgAAAAADdwP0AAAACAACAAAAAAAAAAEAAAASAXYAGQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnEAQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQ
AAAAEAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABCAAAABAAEAAEAAOcQ//8AAOcA//8AAAABAAQ
AAAABAAIAAwAEAAUABgAHAAgACQAKAAsADAANAA4ADwAQABEAAAAAAAABagKGA6YFIAYkBzIIHgk
ACpAMFA1SDsAQfBEYEqYTqBUwAAAAGQAAAAADdwP0ABEAIwA1AFQAawCSAJYAmgCeAKIApgCqAK4
AsgC2ALoAvgDCAMYAygDOANIA1gDjASEAACUzPwcvByU7AT8GLwYjBTsBPwYvBiMlFzMfCRUPDCM
VIzUjHwIzPwEzFSM1NyMPASMvASMfASM1IxczHwUdAQ8GFR8GHQEPCTU3MzUjOwE1IwczNSMhMzU
jBzM1IwczNSM7ATUjBzM1IzczNSMHMzUjFzM1IwUzNSMhMzUjBTM1IzsBNSMHMzUjFzM1IzcXESE
RPwcHER0BHw0zITM/DTURLwghIw8NASEWBgYFBAQCAQEBAQIEBAUGHAGQDwgGBgQDAwEBAQEDAwQ
GBhf+cBIFBgQEAwIBAQEBAgMEBQYWAaYICA0MBQQEAwMCAgIBAQIDAwQFBAYGBgYOGye2JAQCAQM
nOScBAQQqICoGAQEBJGAHBg0KCQcFAwIDBAUGBggJCAcGBQQCAgICAwQICwsOSsicnJw+Pts/PwE
ZICC7Xl59Hx/aICB8Hx98ICB8Hx/6Hx/+iCAgAVggIP7IHx/6Pj7bPz8/nJzsqv1QAQMEBgcECAl
JAgICAwQFBQUGBwcHBwgIAlIICAcHBwcGBQUFBAMCAgIBAQMEqQUGBgb+KQgIBwcHBwYFBQUEAwI
CAmsBAQIDAwQFBQUEBAQCAgEBDwECBAMFBgYHBQUDAwMBKAECAwMEBAUFBAMDAgEBHgEDBAMDBAQ
FBQUNDQYGBQUFBQQEAwMCAgI9sGgNDA10sGkmE3x2GRl2sAECBAQGBwgKBgcGBQUEBAIBAQMEBgY
GCAcGCwUFBAQHBQQBAa/FHx8fHz4fID8+ICAgPiAgIH6dnZ0+Pj4gICABIH2y/iECYgkJCAcGAgQ
CL/2e3BYIBwcHBwYFBQUEAwICAgICAgMEBQUFBgcHBwcICALJBgUFBbIEAwIBAgICAwQFBQUGBwc
HBwgAAAoAAAAAA3cD9AASABgAKwB+AIIAhgCKAJYAowDhAAAlHwIzPwEzBxcjLwEjDwEjNy8BFTM
VIzUjHwIxPwEzBxcjLwEjDwEjNyclHwEVLwMjDwUdAR8LFQ8KLwM1HwMzPwYvDD8JOwEBITUhNzM
1IzUzNSMPATM3FzMnNyMHJyMlFxEhET8HBxEdAR8NMyEzPw01ES8IISMPDQKyGAICAQUaKjI0LRw
DAQQcLTUxyj1ifBgDAgYaKjIzLB0DAQMdLDQwAXEKCQUJCwoLBQUEAwMBAgIEChYJBwYFBAIBAQI
CAwMEBAoMDA0NDQsKBQsMDAwGBQQDAgEBAQEEBQUdCAcFBAMCAQEBAQMCBAgJDAwMDf7SAZf+abz
b29vbhjYrIyMrNjUrIiIqAYeq/VABAwQGBwQICUkCAgIDBAUFBQYHBwcHCAgCUggIBwcHBwYFBQU
EAwICAgEBAwSpBQYGBv4pCAgHBwcHBgUFBQQDAgIC/jUHCA80Vlk5DAw5WFcBkCCwNQcIDzRWWTk
MDDlYVwICAyUDBQMBAQEDAgMEAwQEBAMGCwQGBgYICAoHBwYFBQQEAwUEAgEBAgMEJwQHBAIBAQI
DAwMEBQIGBAMOBQYGBgcICAcGBgUFBAcGAwMBAB8/Hz8fTk83N09ONjb6sv4hAmIJCQgHBgIEAi/
9ntwWCAcHBwcGBQUFBAMCAgICAgIDBAUFBQYHBwcHCAgCyQYFBQWyBAMCAQICAgMEBQUFBgcHBwc
IAAAJAAAAAAN3A/QABwALABcAHwAiAGMApACxAO8AACUHMy8CIwclFSM1Ix8CMz8CMwcjJyMXIyc
jByM3EzcnFxUPDisBLw4/Dx8OBRcVHw4/Dy8PDw4BFxEhET8HBxEdAR8NMyEzPw01ES8IISMPDQF
4Ey0TAgEBAQEvKI8lAgEBAQMkKTwtOzQ+Kg0+DSo/XZyc2gECBAYHCQoLDA4ODxAQERIREREPDw8
NDAsKCQcGBAMBAQMEBgcJCgsMDQ8PDxERERIREBAPDg4MCwoJBwYEAv6KAQMFBwkKDA0OEBERExQ
UFBUUFBISERAODQwKCQYFAwICAwUGCQoMDQ4QERISFBQVFBQUExEREA4NDAoJBwUDAYiq/VABAwQ
GBwQICUkCAgIDBAUFBQYHBwcHCAgCUggIBwcHBwYFBQUEAwICAgEBAwSpBQYGBv4pCAgHBwcHBgU
FBQQDAgICzzs6CAgHJ7CwegoICQl6sLCwJyewAQJtbm4ICRERDw8PDQwLCgkHBgQDAwQGBwkKCww
NDw8PERERERhEQEA8ODgwLCgkHBgQCAQECBAYHCQoLDA4ODxAQERIKChQUExEREA4NDAoJBwUDAQE
DBQcJCgwNDhARERMUFBQVFBQSEhEQDg0MCgkGBQMBAQMFBgkKDA0OEBESEhQUAVOy/iECYgkJCAc
GAgQCL/2e3BYIBwcHBwYFBQUEAwICAgICAgMEBQUFBgcHBwcICALJBgUFBbIEAwIBAgICAwQFBQU
```

```
GBwcHBwgAAAALAAAAAAN3A/QAEgAzAFYAmADYANwA4ADkAPEA/gE8AAAlMz8HNS8GIxcPBB8HPwY
9AS8GDwInHw4dAQ8OIzUFFS8CIw8HHwc/AxUPAy8OPQE/DTsBHwEHHwcPDy8PPw47AR8FASE1ITc
zNSM1MzUjBzM3FzM3IwcnIwcnIyUXESERPwcHER0BHw0zITM/DTURLwghIw8NASEUBgwKCAgFBAE
BBAUICAoMGrwDAgQCAQECBAUHBwkJCggIBgYDAwMDBQcHCQkKCQesCwsKCQgIBwYFBQQDAwIBAQI
DBAQFBgcHCAgJCQoKPgHzCQkLCwsLCQgHBQMBAQMEBwcJCwsLCwsJCgwNDwoJCQgHCAYGBgQEAwM
CAQECAwQEBgYGCAcICQkKCgwMCqgFBQQDAwEBAQEBAgIEBAUFBgcHCAgICQoJCQgIBwcHBgYEBQM
CAgEBAQECAgQEBQUHBggHCAkJCgkJCAgHBwf+7wGX/mnbvLy8vLshHBwhJiYUGxwcEyYBj6r9UAE
DBAYHBAgJSQICAgMEBQUFBgcHBwcICAJSCAgHBwcHBgUFBQQDAgICAQEDBKkFBgYG/ikICAcHBwc
GBQUFBAMCAgJuAQIDBggJCgwNCwoIBwYDAg4EBQkLDA0LCQgHBQMBAQMFBggKCwwNCwoIBwQDAQE
CBSgBAQICAwQEBQYGBwgICQkLCgkJCQcIBwYGBQQDAwIBAbAEJgQEAgECBQcJCQwMDAsJCQYFAwE
BAgQFJAQDAgEBAQICBAQEBgYHBwgICAkKCgoJCQgIBwcGBgQEAwIBAQIVBwYICAgJCQoKCgkICAg
HBgYFBQMDAgEBAQECAgQEBQYGBwcICAkJCgoJCQkICAcHBgUEBAMCAQECAwMEBQESHz8fPx+VXl6
VX19fX/qy/iECYgkJCAcGAgQCL/2e3BYIBwcHBwYFBQUEAwICAgICAgMEBQUFBgcHBwcICALJBgU
FBbIEAwIBAgICAwQFBQUGBwcHBwgAAAAHAAAAAAN3A/QACwAeACoAVgCGAJMA0QAAJRUjFTMVIxU
zFSM1Ix8CMz8BMwcXIy8BMQ8BIzcvARUjFTMVIxUzFSM1ARUPCS8JPQE/CR8JJw8CJwcXDwIjFTM
fAgcXNx8CFTM1PwIXNyc/AjM1Iy8CNycHLwI1IzcXESERPwcHER0BHw0zITM/DTURLwghIw8NAss
9OTlBaYIZAwIBBRwsNDYvHgQEHy84Mxc9OTlCagEGAQEFBwgKCwYHBgYHBgsKCAcFAQEBAQUHCAo
LBgcGBgcGCwoIBwUBAV0JCA8vLTAIAwJDQwIDCDAtLw8ICT4JCA8vLTAIAwNDQwMDCDAtLw8ICT7
Nqv1QAQMEBgcECAlJAgICAwQFBQUGBwcHBwgIAlIICAcHBwcGBQUFBAMCAgIBAQMEqQUGBgb+KQg
IBwcHBwYFBQUEAwICAv4hJyAoILA1BwgPNFZZOQwMOVhXASEnICggsAF/BgcGCwoIBwUBAQEBAQE
FBwgKCwYHBgYHBgsKCAcFAQEBAQEBBQcICgsGB3MDAwgwLDAPCAk+CQgPLy0wCAMDQkIDAwgwLS8
PCAk+CQgPMCwwCAMDQp2y/iECYgkJCAcGAgQCL/2e3BYIBwcHBwYFBQUEAwICAgICAgMEBQUFBgc
HBwcICALJBgUFBbIEAwIBAgICAwQFBQUGBwcHBwgADQAAAAADdwP0AAQAFgAmAEUAXABgAGUAagB
uAHMAkQCeANwAACUPATM1BzsBPwYvBiM3FTMVIxUjNSM1PwUjFzMfCRUPDCMVIzUjHwIzPwEzFSM
1NyMPASMvASMXFSM1ARUjNSMdASM1JR0BIzUlFSM1Ix0BIzUnFREzNTMVMzUzFTM1MxUzESMVIzU
jFSM1IxUjNSMlFxEhET8HBxEdAR8NMyEzPw01ES8IISMPDQLWCxYqrw8IBgYEAwMBAQEBAwMEBgY
X0hUVI0wKFRUJCAaYCAgNDAQFBAMDAgICAQECAwMEBQUFBgYGDxootSQDAwEDJzknAQEEKiAqBgE
CIwGiP/o/ARq8ARo/+j8fID4fvB8/Hx8/H7wfPx8Biar9UAEDBAYHBAgJSQICAgMEBQUFBgcHBwc
ICAJSCAgHBwcHBgUFBQQDAgICAQEDBKkFBgYG/ikICAcHBwcGBQUFBAMCAgK8EB09IwECBAMFBgY
HBQUDAwMBH28dJCQeDBscDw4OAQMEAwMEBAUFBQ0NBgYFBQUFBAQDAwICAj2waA0MDXSwaSYTfHY
ZGXawAWBeXj8fXl4/Xp0fXl4fP14fH/7mICA/PyAgAVgfHz4+Hx+8sv4hAmIJCQgHBgIEAi/9ntw
WCAcHBwcGBQUFBAMCAgICAgIDBAUFBQYHBwcHCAgCyQYFBQWyBAMCAQICAgMEBQUFBgcHBwcIAAA
AEwAAAAADdwP0ABEAMAA0AD4AQgBGAEoATgBSAFYAWgBeAGIAZgBqAG4AcgB/AL0AACU7AT8FPQE
vBSM3FzMfCRUPDCMVIzUjFSM1IxUHMxUjNTcjNRMVIzUHMzUjNzM1KwEzNSM7ATUrATM1IzsBNSs
BMzUjOwE1IwczNSM7ATUrATM1IzsBNSM3FxEhET8HBxEdAR8NMyEzPw01ES8IISMPDQJmEAcGBgQ
EAgICAgQEBgYXFggIDQwFBAQDAwICAgEBAgMDBAQFBgUHBg4bJygnGlhYjVlTsT4gfX0/Pj4/Pz8
/Pj4/Pz8/Pj4/Pz8/Pj4/Pz8/Pj7NqvlQAQMEBgcECAlJAgICAwQFBQUGBwcHBwgIAlIICAcHBwcGBQUFBAMCAgIBAQMEqQUGBgb+KQgIBwcHBwYFBQUEAwICAqkBAgQDBQYGBwUFAwMDAR8
BAwQDAwQEBQUFDQ0GBgUFBQUEBAMDAgICPbCwsBd5IBd4IAEiHx8+Xh8fHyAfHx8gASAfHyAfsv4
hAmIJCQgHBgIEAi/9ntwWCAcHBwcGBQUFBAMCAgICAgIDBAUFBQYHBwcHCAgCyQYFBQWyBAMCAQI
CAgMEBQUFBgcHBwcIAAAAAAkAAAAAA3cD9AAHABkAPABEAEwAUABsAHkAtwAAJQczLwIjBxczPwY
9AS8GNx8JDwcxHwQjLwQjFSM1IxcjJyMHIzcjFSMVIzUjNQMhNSE1IRUzFSMVIRUhNTM1IzUzNSM
1MzUjNTM1IzUhJRcRIRE/BwcRHQEfDTMhMz8NNREvCCEjDw0B+RMtEwIBAQGWEQYFBQQEAgICAgM
EBAYYFwgNDAoIBAIDAwEBAgQEBgcHCQQHBwkhLRsJBgMDDihUPywMPg0qPzMxKDIDARr+5gE5Pj7
+xwFYfT4fHx8fPn3+qAGJqv1QAQMEBgcECAlJAgICAwQFBQUGBwcHBwgIAlIICAcHBwcGBQUFBAM
CAgIBAQMEqQUGBgb+KQgIBwcHBwYFBQUEAwICAs87OggIBygBAQIDBAUFBQUFBAMDAQEBHgEBAwU
GAwQECgsJCAcHBgUEAwIFCAw0LQ0GAgFDsLAnJ7Ahj48gAQPbHz6dPh8fPiAfHx8gPh+8sv4hAmI
JCQgHBgIEAi/9ntwWCAcHBwcGBQUFBAMCAgICAgIDBAUFBQYHBwcHCAgCyQYFBQWyBAMCAQICAgM
EBQUFBgcHBwcIAAAADAAAAAADdwP0ABIAMgBEAGYAqADoAOwA8AD0AQEBDgFMAAA3OwE/BjUvBiM
XDwMfBz8HLwcPAiUfATE/ATMHFyMvASMPASM3JyEzHw0dAQ8NIzUFFS8DDwcfBz8DFQ8DIy8NPQE
/DTsBHwEHHwYdAQ8NKwEvDT0BPw07AR8FAyE1ITczNSM1MzUjBzM3FzM3IwcnIwcnIyUXESERPwc
HER0BHw0zITM/DTURLwghIw8N+xEFCgkIBgUDAgICQoWpAUDAgEBAgMFBgYICAgIBgYFAwI
BAQIDBAYHBwgJCAYBIBYFBRgnLi8pGgMBAxspMS3+cwoKCAgIBgYGBAUDAwICAQECAgQDBQUGBgc
HCAgIQAG1CAgJCgoJCQcFBQIBAQIEBgYICQoKCgkICQoMDQgICAcHBgYEBAUDAgICAQECAgQEBAU
GBwYIBwgJCAsKCZMEBQMDAgIBAQICAwQEBQUGBwcHCAgCyQYFBQWyBAMCAQICAgMEBQUFBgcHBwc
GBwYIBwgJCAcIBgcGBccBl/5p27y8vLy7IRwcISYmFBscHBmAY+q/VABAwQGBwICUkCAgIDBAU
FBQYHBwcHCAgCUggIBwcHBwYFBQUEAwICAgEBAwSpBQYGBv4pCAgHBwcHBgUFBQQDAgICigIDBQc
```

```
ICQoLCggIBgUDAg0HCQkLCwkJBwYEAgEBAgQGBwgKCgwKCAcGBAMBAQMEIy4NDS5MTTIKCzFMTQE
CAgIEAwUFBQcGCAcJCQkICAcHBgYGBQQEAwICAZkDIgUCAgEBAwQGBwkKCgsJCQcGBAIBAQEDBSA
DAwEBAQICAwQEBQUGBgcHCAcJCQgJBwcHBwUGBAQEAgIBAQISBgYGBwcICAkJCAgHBwcGBgUFAwQ
CAgEBAgIDBAQFBgYGBwcICAgJCAgIBwcGBgUFBAMCAgEBAgIDBAQBBx8/Hz8flV5elV9fX1/6sv4
hAmIJCQgHBgIEAi/9ntwWCAcHBwcGBQUFBAMCAgICAgIDBAUFBQYHBwcHCAgCyQYFBQWyBAMCAQI
CAgMEBQUFBgcHBwcIAAAKAAAAAAN3A/QAEQAwAFIAlwCcALwA0QD1AQIBQAAAJTsBPwU9AS8FIzc
7AR8JFQ8MIxUjNSMVDw0jLwI1HwM/Bj0BBRUvAiMPBx8HPwI1IzUzFQ8DLw49AT8NOwEfAQMnBzM
vAR8HPwcvBw8GNzMfAxEPAyEvAxE/AwcRHwcFPwcTLwchIw8FJRcRIRE/BwcRHQEfDTMhMz8NNRE
vCCEjDw0BzxAHBgYEBAICAgIEBAYGFxcIBw4LBQQEAwMDAQMBAgIDAwQEBQYFBgcOGycmAQICAgM
DBAQFBQYGBggOBgcGBgUGBgcFBQMDAwEBcAsLDQ8LCwkJBgUDAQECBQYHCQoLDQUFI0oGDxARFAk
KCAgIBwYGBQQDAwIBAQIDBAUFBgcICAgJCgoLDgwMyV4V+i4gAQEDBAUFBgYHBQYEBAMCAQECAwQ
EBgUHBgYFBQQDAWEDAgMCAQECAwL+2wUDAgEBAgMCKAEDBAUGBwQIASgICAcGBQQCAQEBAwQFBgc
ECP7ZCQgHBgUEAwFpqv1QAQMEBgcECAlJAgICAwQFBQUGBwcHBwgIAlIICAcHBwcGBQUFBAMCAgI
BAQMEqQUGBgb+KQgIBwcHBwYFBQUEAwICAqkBAgMEBQYGBgYFAwMCAh4DBAMDBAQFBQUNDQYGBQU
FBQQEAwMCAgI9r2kJDwcHBgUFBAQDAwIBAQEBAiUDAwEBAQIDBAYHCAloBCUFAwMBAwUHCAoLDQ0
LCggGBQMBAQIBIx9YAwYEAgEBAgIDBAUGBgYIBwkJCQoKCgkICQcIBgYGBAQDAgECAgEkSWmcPwc
FBgQEAwIBAQIDBAQGBQcGBgUFBAMBAQEBAwQFBQY4AQIDAv7bBQMCAQECAwIBJQUDAgEL/t4JCAc
GBQQCAQEBAwQFBgcECAEnCQgHBgUEAgEDBAUGBwi+sv4hAmIJCQgHBgIEAi/9ntwWCAcHBwcGBQU
FBAMDAQICAQMDBAUFBQYHBwcHCAgCyQYFBQWyBAMCAQICAgMEBQUFBgcHBwcIAAAAABMAAAAAA3c
D9AARACEAQACFAIkAjQCRAJUAmQCdAKEApQCpAK0AsQC1ALkAxgEEAAAlOwE/Bi8GIzcfATE1JzM
VIy8BMRUXIzUjOwEfCRUPDCMVIzUFFS8CIw8HHwc/AjUjNTMVDwMvDz8OOwEfASczNSMHMzUjBzM
1IzsBNSMHMzUjOwE1IwczNSMHMzUjOwE1IwczNSM7ATUjBzM1IwczNSMlFxEhET8HBxEdAR8NMyE
zPw01ES8IISMPDQFBDwgGBQUDAwEBAQEDAwUFBhecRgkBJShICQElXAgIDQsFBQQDAwICAgEBAwI
DBAUFBQYGBw4aKAHnCwwNDgwKCggHBAMBAQIFBgcJCgsNBQUjSgcOEBEUCgkICAgHBgYFBAQCAgE
BAQECAwQEBgYHBwgJCQoKCw0NDJc+Pn0+Pn0+Prs/P30/P7w+Pn0+Pn0+Prs/P30/P7w+Pn0+Pn0
+PgFKqv1QAQMEBgcECAlJAgICAwQFBQUGBwcHBwgIAlIICAcHBwcGBQUFBAMCAgIBAQMEqQUGBgb
+KQgIBwcHBwYFBQUEAwICAqkBAgMEBQYGBgYFAwMCAh5qDgN1r24PBXivAwQDAwQEBQUFDQ0GBgU
FBQUEBAMDAgICPa8EJQUDAwEDBQcICgsNDQsKCAYFAwEBAgEjH1gDBgQCAQECAgMEBQYGBggHCQk
JCgoKCQgJBwgGBgYEBAMCAQIC5T4+Pj4+Pz8/Pj4+Pj4/Pz8+Pj4+Pryy/iECYgkJCAcGAgQCL/2
e3BYIBwcHBwYFBQUEAwMBAgIBAwMEBQUFBgcHBwcICALJBgUFBbIEAwIBAgICAwQFBQUGBwcHBwg
AAA0AAAAAA3cD9AAJAA0AUgBkAHYAiACaAKwAvgDQAOIA7wEtAAAlFSMVMxUjFSM1IxUjNQcVLwI
jDwcfBz8CNSM1MxUPAyMvDT0BPw07AR8BNxU/BycPBicfBzUvBzcXPwcjDwYFNy8HIx8GJR8HMy8
HBTM/BycPBjcfBzcvBwcXPwc1DwYlFxEhET8HBxEdAR8NMyEzPw01ES8IISMPDQK+Pjk5JygnKwo
MDQ4MCwkIBwUDAQEDBAYICAoLDQUFI0oGDw8SFAkJCQgIBwYGBQQDAwIBAQMCBAUFBgcICAgKCQo
LDgwMLBAQEA8ODg0MFgoKDAsMDQ2hDQ0ODg8PEBENDQ0MDAsKC/IWCgkIBwUFAwIfAgMEBAYGB/7
aFgcHBwUFAwMCHwIDBQUHCAkBJwgHBgYEBAMCHwIDBQUHCAkK/pwfAgMDBQUHBwcWCggIBwUFA9g
NDQ0MCwwKChYMDQ4ODxAQEJQWCwoLDAwNDQ0REA8PDg4NATWq/VABAwQGBwQICUkCAgIDBAUFBQY
HBwcHCAgCUggIBwcHBwYFBQUEAwICAgEBAwSpBQYGBv4pCAgHBwcHBgUFBQQDAgIC/iErIESwsLA
FJQUDAwEDBQYJCgsNDQsKCAYFAwEBAgEjH1gDBgQBAQICAwQFBgYHBwgICQkKCgoJCQgHCAYGBgQ
EAwIBAgLmIAIDBQUHCAkKFggHBgYEBAMPCQkIBwUFAwIgAQMEBAYGBwgWFgwNDg4PEBAQDQ0NDAs
MCiAWCgoMCwwNDQ0QEBAPDg4N5QoKCwwMDQ0NERAPDw4ODQ11DgwNDAwLCwoWDQ0NDw8PEIsCAwM
FBQcHBxYJCQgHBQUDAjAWBwcHBQUDAwIfAgMFBQcICaWy/iECYgkJCAcGAgQCL/2e3BYIBwcHBwY
FBQUEAwICAgICAgMEBQUFBgcHBwcICALJBgUFBbIEAwIBAgICAwQFBQUGBwcHBwgAAAwAAAAA3c
D9AASACQALgBRAHAAhgCcAKcAvAEqATcBdQAAJTM/BzUvBiMHOwE/Bi8GIyUVIxUzFSMVIzUjMx8
NHQEPDiM1BzsBHwkVDwwjFSM1Nw8GLwc/BSUzHwQdAQ8FLwM/ARcnDwM/Ah8CJxcdAQ8BLwI9AT8
EMx8DJw8JFR8EDw4fCT8MHwY/DTUvCiMPAS8DPwM1Lwk3FxEhET8HBxEdAR8NMyEzPw01ES8IISM
PDQHaFAYLCgkHBgMCAgMGBwkKCxqZDwgGBQUDAwEBAQEDAwUFBhcBjz45OSd6CwsKCQgIBwYFBQQ
DAwIBAQIDBAQFBgcHCAgJCQoKPlwICA0LBQUEAwMCAgIBAQMCAwQFBQUGBgcOGihfBQUGBgMDBAQ
DBAIDAgEBAQIECA4oARoEAwoDAgICAwQFBQUECAgiEREKbRkZGRoPDQkPEBAxAQIKDg8DAQIEBQU
GBwcCAh0GBgYFBQQDAwQBAQQEFRYHCAgKCgsUOwoICAcFAwEBAwQEBgYGBwgHCQgHBgUFBxwRISA
gMRsWDQcHCAUKBgYFBQUFBAQDAgIBAQECAgQCBgcJDBELCxYgGBgWFQcJBAEBAgMFBQcHBwkI2ar
9UAEDBAYHBAgJSQICAgMEBQUFBgcHBwcICAJSCAgHBwcHBgUFBQQDAgICAQEDBKkFBgYG/ikICAc
HBwcGBQUFBAMCAgJuAQIDBggJCgwNCwoIBwYDAjQBAgMEBQYGBgYFAwMCAh4gKyBErwECAgMEBAU
GBgcICAkKCgoJCQkHCAcGBgUEAwMCAQQGwAQMEAwMEBAUFBQ0NBgYFBQUFBAQDAwICAj2v5gkICQU
BAQEBAQIDAwQEBAMFBQYHEEABBAICBAcFBAUEAwIBAQQGHwMBAREFBgcIJCQbExITxAUGBg0mFhk
KBQUFBQMDAQIEAwMrAQIDBAMEBQQKCgsLCgkjIRkYGRgXGAgYBQYGCAkJCQoKBwcFBQQCAgEBAgM
EBQUKMiALCgcKGRQKBQMCAQEBAgIDBAQFBQUFBgYGBgUGBgYDBgUEAwIBAgYZGRsbHCARCQkICQg
```

```
HBgYEBAIBfbL+IQJiCQkIBwYCBAIv/Z7cFggHBwcHBgUFBQQDAwECAgEDAwQFBQUGBwcHBwgIAsk
GBQUFsgQDAgECAgIDBAUFBQYHBwcHCAAAAAcAAAAAA3cD9AAHABoAIgAlAC0AOgB4AAAlFSMVIzU
jNScfAjM/ATMHFyMvASMPASM3LwEVIxUjNSM1EyM3BzM3MxczAyM3FxEhET8HBxEdAR8NMyEzPw0
1ES8IISMPDQLuMicydxkDAgEFHCw1Ni8eAwEEHi84MwkxKDLwQyJ9OhNgEjtkM9Gq/VABAwQGBwQ
ICUkCAgIDBAUFBQYHBwcHCAgCUggIBwcHBwYFBQUEAwICAgEBAwSpBQYGBv4pCAgHBwcHBgUFBQQ
DAgIC/iGPjyABNQcIDzRWWTkMDDlYVwEhj48gAVtkyDc3AQvWsv4hAmIJCQgHBgIEAi/9ntwWCAc
HBwcGBQUFBAMCAgICAgIDBAUFBQYHBwcHCAgCyQYFBQWyBAMCAQICAgMEBQUFBgcHBwcIAAsAAAA
AA3cD9AARAB0AOwBcAKEApgDGANsA/wEMAUoAACU7AT8FPQEvBSM3FSMVMxUjFTMVIzUjFzMfCQ8
MIxUjNSMVDwwjLwI1HwI7AT8GNQUVLwIjDwcfBz8CNSM1MxUPAyMvDT0BPw07AR8BAycHMy8BHwY
7AT8GLwcPBjczHwMRDwMhLwMRPwMHER8HITM/BhMvByEjDwUlFxEhET8HBxEdAR8NMyEzPw01ES8
IISMPDQGIDwcGBQQDAwEBAwMEBQYWyzo2Nj5jVwgHDAsFBAQDAgMDAQIBAgIDBAQFBQUGBg4ZJSQ
BAwIDAwQEBAUGBQcGDgYGBQUFBQYGBQUDAwIBAQHXCgsMDQsKCQgHBAMBAQMEBQgICQsMBQQhRgY
ODxATCQkIBwcHBgUFBAMDAQICAgIEBAUGBwcHCAkJCgoNDAvxXhX6LiABAQMEBQUGBgcFBgQEAwI
BAQIDBAQGBQcGBgUFBAMBYQMCAwIBAQIDAv7bBQMCAQECAwIoAQMEBQYHBAgBKAgIBwYFBAIBAQE
DBAUGBwQI/tkJCAcGBQQDAWmq/VABAwQGBwQICUkCAgIDBAUFBQYHBwcHCAgCUggIBwcHBwYFBQU
EAwICAgEBAwSpBQYGBv4pCAgHBwcHBgUFBQQDAgICqQECBAMFBgYHBQUDAwMBHyEnICggsAEDBAM
DBAQFBQsNDQYFBQUFBAQDAwICAj2wahEOBwYFBQQEAwMCAQEBAQIlAwMBAgMEBgcICmgFJQUDAwE
DBQYJCgsNDQsKCAYFAwEBAgEjH1gDBgQBAQICAwQFBgYHBwgICQkKCgoJCQgHCAYGBgQEAwIBAgI
BJElpnD8HBQYEBAMCAgMEBAYFBwYGBQUEAwEBAQEDBAUFBjgBAgMC/tsFAwIBAQIDAgElBQMCAQv
+3gkIBwYFBAIBAwQFBgcECAEnCQgHBgUEAgEDBAUGBwi+sv4hAmIJCQgHBgIEAi/9ntwWCAcHBwc
GBQUFBAMCAgICAgIDBAUFBQYHBwcHCAgCyQYFBQWyBAMCAQICAgMEBQUFBgcHBwcIAAAJAAAAAAN
3A/QABQAYAGwAcAB0AHgAhACRAM8AACUVMxUjNSMfAjE/ATMHFyMvATEPASM3JyUfARUvAyMPBh8
MFQ8JKwEvAjUfAzM/Bi8MNT8KMwEhNSE3MzUjNTM1Iw8BMzcXMyc3IwcnIyUXESERPwcHER0BHw0
zITM/DTURLwghIw8NAhNBaIIZAgMGHCw1Ni8eBAQeLzcyAYYKCQQKCwsMBQUFAwICAQECAwMLGAg
IBwUEAQIBAgIEAwQFCgwNDQ8NDAoLBgwNDQYFBAMDAQEBAgQEBh8IBwYEBAEBAQICAwQEBAsLDQ4
G/ncBtv5K29vb29uVNywjIyw3NiwiIisBeKr9UAEDBAYHBAgJSQICAgMEBQUFBgcHBwcICAJSCAg
HBwcHBgUFBQQDAgICAQEDBKkFBgYG/ikICAcHBwcGBQUFBAMCAgLwjh+tNAcHDzNWVzgMDDhWVwE
CAyQDBQMBAQEDAgMDBAQEAwQGCgUFBgcHBAoMBgYFBQQEAwUEAgIDBCcIAwQCAQECAwMDBAUEBAM
EDgUFBgYHCAkGBgYFBAUDBAUEAgEBDR8/Hz8fTk83N09ONjb6sv4hAmIJCQgHBgIEAi/9ntwWCAc
HBwcGBQUFBAMCAgICAgIDBAUFBQYHBwcHCAgCyQYFBQWyBAMCAQICAgMEBQUFBgcHBwcIAAAHAAA
AAAN3A/QAEQAwAEcAoAD+AQsBSQAAJTsBPwYvBiM3FzMfCRUPDCMVIzUjHwIzPwEzFSM1NyMPASM
vASMfASM1BR8EDwcVHwcPCy8DNR8DMz8FPQEvBSM1Mz8FPQEvBSMPAzU/AjsBHwEDDwUVLwEPDB8
KPwo1NxUvAQ8MFR8JPwo1ETUvBTcXESERPwcHER0BHw0zITM/DTURLwghIw8NAjAPCAYGBAMDAQE
BAQMDBAYGFxYICA0MBAUEAwMCAgIBAQIDAwQFBQUGBgYPGii1JAMDAQMnOScBAQQqICoGAQEBJAH
fAwMFAwEBAgIEBQYHCAkHBwYFAwIBAQICAgMEBAUFCw4PDQsLCQUJCgsNBgYEBAICAgMFBQcHGRc
HBgUDAwICAgIEBQUGCQgJCAkLCwwMDAlr6wMGBAIBCAkICQkIBgsKCQcCAgEBAQEBBAUFBwcICQo
ICQgJDAoJCAQEAcMICQkICQgGDAkJBwICAQECAQQFBQcHCAkKCAkICQwKCQgEAwICAgMDAwcmqv1
QAQMEBgcECAlJAgICAwQFBQUGBwcHBwgIAlIICAcHBwcGBQUFBAMCAgIBAQMEqQUGBgb+KQgIBwc
HBwYFBQUEAwICAqkBAgQDBQYGBwUFAwMDAR8BAwQDAwQEBQUFDQ0GBgUFBQUEBAMDAgICPbBoDQw
NdLBpJhN8dhkZdrAKAwMICAkICAYGBQQEAgEBAwQFBgcHCAsGBQQFBAMEAwQDAQEBAwQiAwUDAgE
CAwQEBQUFBQQEAwIBHgECAwMEBAYFBAQDAgIBAQIDBR8EAwIDAwJdLwEEBQcD+AEBAQEDAwMHCAo
LBgcGBgYGBggGBgUEAgIBAQECBAYICQoICQnVJ5wBAQEBAgMDBwkJDAYGBwYGBgYHBwYFAwMCAQE
BAwMGCAkKCAkICAEuBQQEAwICAX2y/iECYgkJCAcGAgQCL/2e3BYIBwcHBwYFBQUEAwICAgICAgM
EBQUFBgcHBwcICALJBgUFBbIEAwIBAgICAwQFBQUGBwcHBwgAAAAAAAASAN4AAQAAAAAAAAABAAA
AAQAAAAAAAQAPAAEAAQAAAAAAAgAHABAAAQAAAAAAAwAPABcAAQAAAAAABAAPACYAAQAAAAAABQA
LADUAAQAAAAAABgAPAEAAAQAAAAAACgAsAE8AAQAAAAAACwASAHsAAwABBAkAAAACAI0AAwABBAk
AAQAeAI8AAwABBAkAAgAOAK0AAwABBAkAAwAeALsAAwABBAkABAAeANkAAwABBAkABQAWAPcAAwA
BBAkABgAeAQ0AAwABBAkACgBYASsAAwABBAkACwAkAYMgRmlsZXR5cGVWMiBGb250UmVndWxhckZ
pbGV0eXBlVjIgRm9udEZpbGV0eXBlVjIgRm9udFZlcnNpb24gMS4wRmlsZXR5cGVWMiBGb250Rm9
udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5
jb20AIABGAGkAbABlAHQAeQBwAGUAVgAyACAARgBvAG4AdABSAGUAZwB1AGwAYQByAEYAaQBsAGU
AdAB5AHAAZQBWADIAIABGAG8AbgB0AEYAaQBsAGUAdAB5AHAAZQBWADIAIABGAG8AbgB0AFYAZQB
yAHMAaQBvAG4AIAAxAC4AMABGAGkAbABlAHQAeQBwAGUAVgAyACAARgBvAG4AdABGAG8AbgB0ACA
AZwBlAG4AZQByAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB
lAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0
AAAAAAgAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAASAQIBAwEEAQUBBgEHAQgBCQEKAQs
```

```
BDAENAQ4BDwEQAREBEgETAANCTVAEWExTWANBVkkDRE9DA0VYRQNNUDQDWklQA1RBUgRET0NYA0p
QRwNQTkcDR0lGA1BERgNUWFQESlBFRwNYTFMDTVAzAAA=) format('truetype');
        font-weight: normal;
        font-style: normal;
    }
    [class^="sf-icon-"], [class*=" sf-icon-"] {
        font-family: 'FiletypeV2 Font' !important;
        speak: none;
        font-size: 35px;
        font-style: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
        line-height: 1;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
        position: relative;
        top: 15px;
    }
    .customdroparea .ul-element {
        list-style: none;
        width: 100%;
        padding-left: 0;
    }
    .customdroparea span.list-wrapper {
        max-height: 400px;
        display: block;
        margin-left: 10px;
        position: relative;
        top: 2px;
    }
    .customdroparea .file-name {
        padding: 8px 6px 5px 12px;
        font-size: 13px;
        width: 76%;
        display: inline-block;
        position: relative;
    }
    .customdroparea .file-size {
        padding: 4px;
        font-size: 13px;
        width: 18%;
        display: inline-block;
        position: relative;
    }
    .customdroparea .file-lists {
        border: 1px solid lightgray;
        padding: 0 6px 0 14px;
        margin-top: 15px;
        position: relative;
        background: rgba(0, 0, 0, 0.04);
    }
    .customdroparea .file-size, .customdroparea .file-name {
        text-overflow: ellipsis;
        overflow: hidden;
        white-space: nowrap;
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
/* csslint ignore:end */
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/filelist-template-cs1" %}

Custom template

You can design the own template by preventing the default file list including buttons.

when you use custom template to upload or remove the files, pass the custom UI argument as true to call

upload / remove public method as follows:

- UploaderObj.upload(filesData, true);
- UploaderObj.remove(filesData, true);

Refer to the following code sample.

APP.VUE

```
<template>
    <div class="col-lg-8 control-section uploader template-view">
        <div class="control_wrapper">
            <div class="sample_wrapper">
                <div id="dropTarget">
                    <span id="dropElement" class="droparea">Drop files here
or <a href="" id="browse"><u>Browse</u></a> </span>
                    <ejs-uploader id='template' name="UploadFiles"
:asyncSettings= "path" ref="uploadObj"
                        :dropArea= "dropElement" :selected= "onFileSelect"
:progress= "onFileUpload"
                        :success= "onUploadSuccess" :failure=
"onUploadFailed" :removing= "onFileRemove">
                    </ejs-uploader>
                </div>
            <div style="margin-left: 50px; padding-top:25px;">
            <button class="e-btn e-css" id="clearbtn">Clear All</button>
        </div>
            </div>
        </div>
    </div>
</template>
<script>
import Vue from "vue";
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
import { FileInfo } from '@syncfusion/ej2-vue-inputs/uploader';
import { createSpinner, showSpinner, hideSpinner } from '@syncfusion/ej2-
popups';
import { createElement, isNullOrUndefined, detach, EventHandler } from
'@syncfusion/ej2-base';
Vue.use(UploaderPlugin);
export default {
   data: function() {
        return {
          path:  {
```

```
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
          dropElement: '#dropTarget',
          filesList: [],
          filesDetails: [],
          parentElement: '',
          progressbarContainer: ''
        }
    },
    mounted: function () {
        document.getElementById('browse').onclick = function() {
        document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
        return false;
        };
        document.getElementById('clearbtn').onclick = () => {

if(!isNullOrUndefined(document.getElementById('dropTarget').querySelector('.
upload-list-root'))) {
                this.$refs.uploadObj.$el.value = '';

detach(document.getElementById('dropTarget').querySelector('.upload-list-
root'));
                this.$refs.uploadObj.filesDetails=[];
                this.$refs.uploadObj.filesList=[];
            }
        };
    },
    methods:{
        onFileSelect: function (args) {
            if
(isNullOrUndefined(document.getElementById('dropTarget').querySelector('.upl
oad-list-root'))) {
                this.parentElement = createElement('div', { className:
'upload-list-root' });
                this.parentElement.appendChild(createElement('ul',
{className: 'ul-element' }));

document.getElementById('dropTarget').appendChild(this.parentElement);
            }
            for (let i = 0; i < args.filesData.length; i++) {
                this.formSelectedData(args.filesData[i]);
            }
            this.filesDetails = this.filesDetails.concat(args.filesData);
            this.$refs.uploadObj.upload(args.filesData, true);
            args.cancel = true;
        },
        formSelectedData: function ( selectedFiles) {
            let liEle = createElement('li',  { className: 'file-lists',
attrs: {'data-file-name' : selectedFiles.name} });
            liEle.appendChild(createElement('span', {className: 'file-name
', innerHTML: selectedFiles.name }));
            liEle.appendChild(createElement('span', {className: 'file-size
', innerHTML: this.$refs.uploadObj.bytesToSize(selectedFiles.size) }));
```

Copyright © 2001 -2024 Syncfusion Inc.

```
            if (selectedFiles.statusCode === '1') {
                this.progressbarContainer = createElement('span',
{className: 'progress-bar-container'});

this.progressbarContainer.appendChild(createElement('progress', {className:
'progress', attrs: {value : '0', max : '100'}} ));
                liEle.appendChild(this.progressbarContainer);
            } else { liEle.querySelector('.file-
name').classList.add('upload-fails'); }
            let closeIconContainer = createElement('span', {className: 'e-
icons close-icon-container'});
            let localObj = this;
            closeIconContainer.addEventListener( 'click', function(e) {
                localObj.removeFiles(e);
            });
            liEle.appendChild(closeIconContainer);
document.querySelector('.ul-element').appendChild(liEle);
            this.filesList.push(liEle);
        },
        onFileUpload: function(args) {
            let li =
document.getElementById('dropTarget').querySelector('[data-file-name="' +
args.file.name + '"]');
            let localObj = this;
            let progressValue = Math.round((args.e.loaded / args.e.total) *
100);
            if (!isNaN(progressValue)) {
                li.getElementsByTagName('progress')[0].value =
progressValue;
            }
        },
        onUploadSuccess: function(args) {
            let spinnerElement = document.getElementById('dropTarget');
            let li=
document.getElementById('dropTarget').querySelector('[data-file-name="' +
args.file.name + '"]');
            if (!isNullOrUndefined(li.querySelector('.progress-bar-
container'))) {
                detach(li.querySelector('.progress-bar-container'));
            }
            let localObj = this;
            if (args.operation === 'upload') {
                li.querySelector('.file-name').classList.add('upload-
success');
                li.querySelector('.close-icon-
container').classList.remove('remove-btn');
                li.querySelector('.close-icon-
container').classList.add('delete-icon');
                (li.querySelector('.close-icon-container')).onclick =
function() {
                    localObj.generateSpinner(li.querySelector('.close-icon-
container'));
                };
                (li.querySelector('.close-icon-container')).onkeydown =
function(e) {
                    if (e.keyCode === 13) {
```

```
                        localObj.generateSpinner(e.target.closest('.e-
upload'));
                    }
                };
            }
            if (args.operation === 'remove') {
                this.filesDetails.splice(this.filesList.indexOf(li), 1);
                this.filesList.splice(this.filesList.indexOf(li), 1);
                detach(li);
                hideSpinner(li.querySelector('.close-icon-container'));
                detach(li.querySelector('.e-spinner-pane'));
            }
        },
        generateSpinner: function(targetElement) {
            createSpinner({ target: targetElement, width: '25px' });
            showSpinner(targetElement);
        },
        onUploadFailed: function(args) {
            let li =
document.getElementById('dropTarget').querySelector('[data-file-name="' +
args.file.name + '"]');
            let localObj = this;
            li.querySelector('.file-name').classList.add('upload-fails');
            li.querySelector('.close-icon-
container').classList.remove('remove-btn');
            if (args.operation === 'remove') {
                if (!isNullOrUndefined(li)) {
                    this.filesDetails.splice(this.filesList.indexOf(li), 1);
                    this.filesList.splice(this.filesList.indexOf(li), 1);
                    detach(li);
                }
            }
            if (args.operation === 'upload') {
                detach(li.querySelector('.progress-bar-container'));
            }
        },
        removeFiles: function(args) {
            if (!isNullOrUndefined(args.currentTarget)) {
                if
(this.filesDetails[this.filesList.indexOf(args.currentTarget.parentElement)]
.statusCode === '2' ) {

this.$refs.uploadObj.remove(this.filesDetails[this.filesList.indexOf(args.cu
rrentTarget.parentElement)]);
                } else  {
                    this.onFileRemove(args);
                }
            }
        },
        onFileRemove: function(args) {
            args.postRawFile = false;
            if (!isNullOrUndefined(args.currentTarget)) {
                if
(this.filesDetails[this.filesList.indexOf(args.currentTarget.parentElement)]
.statusCode !== '2') {
                    detach(args.currentTarget.parentElement);
```

Copyright © 2001 -2024 Syncfusion Inc.

```
this.filesList.splice(this.filesList.indexOf(args.currentTarget.parentElemen
t), 1);
                }
            }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
     #dropTarget {
            min-height: 50px;
            padding-top: 15px;
            position: relative;
    }
    #clearbtn {
        margin: 0px 0px 0px 30%;
    }
    #dropElement {
        padding: 3% 28% 3%;
        display: inherit;
        border: 1px dashed #c3c3cc
    }
    .template-view .control_wrapper {
        max-width: 400px;
        margin: auto;
    }
    .template-view .e-file-select-wrap {
        display: none;
    }
    .template-view .e-upload {
        float: none;
        border: none;
    }
    .template-view .ul-element {
        list-style: none;
        width: 100%;
        padding-left: 0;
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    .template-view .file-name {
        padding: 8px 6px 8px 0;
        font-size: 13px;
        width: 46%;
        display: inline-block;
        position: relative;
        top: 4px;
    }
    .template-view .file-size {
        padding: 4px;
        font-size: 13px;
        width: 18%;
        display: inline-block;
        position: relative;
    }
    .template-view .file-lists {
        border: 1px solid lightgray;
        padding: 0 6px 0 14px;
        margin-top: 15px;
        position: relative;
        background: rgba(0, 0, 0, 0.04);
    }
    .template-view .file-size, .template-view .file-name {
        font-family: "Helvetica Neue", "Helvetica", "Arial", "sans-serif";
        text-overflow: ellipsis;
        overflow: hidden;
        white-space: nowrap;
    }
    .template-view span.progress-bar-container {
        display: block;
        float: right;
        height: 20px;
        right: 13%;
        top: 14px;
        position: relative;
        width: 20%;
    }
    .template-view .progress{
        width: 100%;
        height: 15px;
        -webkit-appearance: none;
    }
    .template-view .close-icon-container
    {
        cursor: pointer;
        font-size: 11px;
        height: 24px;
        margin: 0 12px 0 22px;
        padding: 0;
        position: absolute;
        right: 0;
        width: 24px;
        top: 6px;
    }
    .template-view .close-icon-container.remove-btn {
       display: none;
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    .template-view .close-icon-container.e-icons::before {
        left: 7px;
        position: inherit;
        top: 7px;
        content: '\e932';
    }
    .template-view .close-icon-container.delete-icon::before {
        content: '\e94a';
    }
    .template-view .close-icon-container:hover {
        background-color: rgba(0, 0, 0, 0.12);
        border-color: transparent;
        border-radius: 50%;
        box-shadow: 0 0 0 transparent;
    }
    .template-view .upload-success {
       color: #2bc700;
    }
    .template-view .upload-fails {
        color: #f44336;
    }
    .template-view progress::-webkit-progress-bar {
        border: 1px solid lightgrey;
        background-color: #ffffff;
        border-radius: 2px;
    }
    #dropTarget progress {
        border: 1px solid lightgrey;
        background-color: #ffffff;
        border-radius: 2px;
    }
    .material .template-view progress::-webkit-progress-value {
        border-radius: 2px;
        background-color: #ff4081;
    }
    .material .template-view progress::-moz-progress-bar {
        border-radius: 2px;
        background-color: #ff4081;
    }
    .material #dropTarget span a {
        color:#ff4081;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/custom-template-cs1" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

See Also

- Customize progress bar
- Customize button with HTML element
- Customize drop area

Copyright © 2001 -2024 Syncfusion Inc.

Localization in Vue Uploader component

The Localization library allows you to localize static text content of the uploader. The static text contains default text content of action buttons, file status, clear icon title, tooltips, and text content of drag area. Define the locale object for a culture and assign it to L10n load method.

The following are the list of keys and its values used in the uploader.

| Key | Description |

|----------|---------|

| Browse | To customize the browse button text.|

| Clear | To customize the clear button text.|

| Upload | To customize the upload button text. |

| dropFilesHint | To customize the drop area text. |

| uploadFailedMessage | To customize the status text when the file is failed to upload.|

| uploadSuccessMessage | To customize the status text when the file is uploaded successfully.|

| removedSuccessMessage | To customize the status text when the file is removed the successfully from the server.|

| removedFailedMessage | To customize the status text while the file is failed to remove.|

| inProgress | To customize the status text while the upload is in progress.|

| pauseUpload | To customize the status text while the uploading is paused.|

| fileUploadCancel | To customize the status text when uploading is cancelled.|

| readyToUploadMessage | To customize the status text when the file is selected and ready to upload.|

| invalidMaxFileSize | To customize the status text when the file size is greater than the maximum file size.|

| invalidFileType | To customize the status text when the file type is invalid.|

| invalidMinFileSize | To customize the status text when the file size is less than the minimum file size. |

| remove | To customize tooltip text for remove icon. |

| cancel | To customize tooltip text for cancel icon. |

| delete | To customize tooltip text for delete icon. |

| totalFiles | To customize tooltip text for total files. |

| size | To customize tooltip text for size. |

APP.VUE

```
<template>
    <div class="col-lg-8 control-section uploader chunk">
        <div class="control_wrapper">
            <ejs-uploader id='validation' name="UploadFiles" :asyncSettings=
"path" ref="uploadObj" :locale='locale' :autoUpload = 'false'>
            </ejs-uploader>
        </div>
    </div>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from "vue";
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
import { FileInfo } from '@syncfusion/ej2-vue-inputs/uploader';
import { detach, L10n } from '@syncfusion/ej2-base';
Vue.use(UploaderPlugin);
    L10n.load({
    "fr-CH": {
        "uploader": {
            "invalidMinFileSize" : "La taille du fichier est trop petite!
S'il vous plaît télécharger des fichiers avec une taille minimale de 10 Ko",
            "invalidMaxFileSize" : "La taille du fichier dépasse 28 Mo",
            "invalidFileType" : "Le type de fichier n'est pas autorisé",
            "Browse"  : "Feuilleter",
            "Clear" : "Clair",
            "Upload" : "Télécharger",
            "dropFilesHint" : "ou Déposer des fichiers ici",
            "uploadFailedMessage" : "Impossible d'importer le fichier",
            "uploadSuccessMessage" : "Fichier téléchargé avec succès",
            "removedSuccessMessage": "Fichier supprimé avec succès",
            "removedFailedMessage": "Le fichier n'a pas pu être supprimé",
            "inProgress": "Téléchargement",
            "readyToUploadMessage": "Prêt à télécharger",
            "remove": "Retirer",
            "cancel": "Annuler",
            "delete": "Supprimer le fichier",
            "totalFiles": "Total des fichiers",
            "size": "taille"
        }
    }
})
export default {
 data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
          extensions: '.doc, .docx, .xls, .xlsx',
          locale: 'fr-CH'
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
    #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
```

```
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/localization-cs1" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Wai aria accessibility in Vue Uploader component

The Uploader component followed the accessibility guidelines and standards, including ADA, Section 508, WCAG 2.2 standards, and WCAG roles that are commonly used to evaluate accessibility.

The accessibility compliance for the Uploader component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Section 508 Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Screen Reader Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Right-To-Left Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Color Contrast | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Mobile Device Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Keyboard Navigation Support | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Accessibility Checker Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

| Axe-core Accessibility Validation | <img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<img src="https://cdn.syncfusion.com/content/images/landing-page/yes.png" alt="Yes"> - All features of the component meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/intermediate.png" alt="Intermediate"> - Some features of the component do not meet the requirement.

<img src="https://cdn.syncfusion.com/content/images/landing-page/no.png" alt="No"> - The component does not meet the requirement.

Keyboard interaction

The uploader component characterized with complete ARIA accessibility support that helps to be accessible by on-screen readers and other assistive technology devices.

The following are the standard keys that works on uploader component:

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Tab | Move focus to next element. |

| Shift + Tab | Move focus to previous element. |

| Enter | Triggers corresponding action to button element. |

| Esc | Close the file browser dialog alone and cancels the upload on drop the file.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" ></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
    data: function() {
        return {
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            }
        }
    }
}
</script>
```

Copyright © 2001 -2024 Syncfusion Inc.

```
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/accessibility-cs1" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Ensuring accessibility

The Uploader component's accessibility levels are ensured through an accessibility-checker and axe-core software tools during automated testing.

The accessibility compliance of the Uploader component is shown in the following sample. Open the sample in a new window to evaluate the accessibility of the Uploader component with accessibility tools.

{% previewsample "https://ej2.syncfusion.com/accessibility/uploader.html" %}

See also

- Accessibility in Syncfusion Vue components

Style appearance in Vue Uploader component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of File Upload wrapper element

Use the following CSS to customize the appearance of wrapper element.

`

/ *To specify height* /

.e-upload.e-control-wrapper, .e-bigger.e-small .e-upload.e-control-wrapper {

height: 300px;

width: 300px;

}

`

Customizing the File Upload browse button

Use the following CSS to customize the File Upload browse button

`

/ *To specify font size and color* /

.e-upload .e-file-select-wrap .e-btn, .e-upload .e-upload-actions .e-btn, .e-bigger.e-small .e-upload .e-file-select-wrap .e-btn, .e-bigger.e-small .e-upload .e-upload-actions .e-btn {

Copyright © 2001 -2024 Syncfusion Inc.

font-family: cursive;

height: 40px;

background-color: aquamarine;

color: coral;

}

`

Customizing the File Upload content

Use the following CSS to customize the File Upload content

`

/ *To specify font size and color* /

.e-upload .e-file-select-wrap .e-file-drop, .e-bigger.e-small .e-upload .e-file-select-wrap .e-file-drop {

font-size: 20px;

color: aqua;

}

`

Customizing the uploaded file container in File Upload

Use the following CSS to customize the uploaded file container in File Upload

`

/ *To specify background color* /

.e-upload .e-upload-files .e-upload-file-list {

background-color: beige;

}

`

See Also

- Customize the appearance of uploader using a template

How To

Hide default drop area in Vue Uploader component

You can achieve this behavior by override the corresponding uploader styles. In the following example, override the below styles to hide the default drop area behavior.

- .e-upload.e-control
- .e-upload .e-file-select
- .e-upload .e-file-drop

APP.VUE

Copyright © 2001 -2024 Syncfusion Inc.

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path" ></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
    data: function() {
        return {
            path:  {
                saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
                removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
            }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .e-control .e-file-select,
    .e-control .e-file-drop {
        display: none;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/hide-drop-cs1" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Copyright © 2001 -2024 Syncfusion Inc.

Preview images before uploading in Vue Uploader component

The uploader component allows to create preview images before uploaded it. The preview images created by reading the file using selected event.

Also, the user can create preview images after uploading to server using success event.

In the following link, explains about how to create image preview in uploader component.

Image Preview

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Achieve invisible upload in Vue Uploader component

You can achieve the invisible upload feature by using selected event in uploader component. In the following example, explains how to achieve invisible upload in uploader component.

APP.VUE

```
<template>
  <div>
  <div id='preview'></div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:selected = "onupload" :autoUpload= "autoUpload" :asyncSettings= "path"
:allowedExtensions = "extensions" ></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin, SelectedEventArgs } from '@syncfusion/ej2-vue-
inputs';
import { createElement, detach, EventHandler } from '@syncfusion/ej2-base';
Vue.use(UploaderPlugin);
export default {
  data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
          extensions: '.png, .jpg, .jpeg',
          autoUpload: false
        }
    },
    methods: {
        onupload:function(args: SelectedEventArgs){
    for(let i = 0; i< args.filesData.length ; i++){
        let liparentDiv = createElement('div',  { className: 'image-list'});
        let liImage = createElement('img',  { className: 'image'});
        liparentDiv.appendChild(liImage);
        this.readURL(liImage, args.filesData[i]);
        document.getElementById('preview').appendChild(liparentDiv);
    }
    args.cancel=true;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
 readURL: function(liImage: HTMLElement, file: any) {
    let imgPreview: HTMLImageElement = liImage as HTMLImageElement;
    let imageFile: File = file.rawFile;
    let reader: FileReader = new FileReader();
    reader.addEventListener( 'load', () => {
        imgPreview.src = reader.result;
    }, false);
    if (imageFile) {
        reader.readAsDataURL(imageFile);
    }
};
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    #preview {
  border: 2px dashed #ddd;
  padding: 15px;
}
.image-list {
  width: 134px;
  height: 117px;
  margin-right: 4px;
  border: 1px solid lightgrey;
  display: inline-block;
}
.image {
  width: 134px;
  height: 117px;
  margin-right: 4px;
  display: inline-block;
}
.e-control {
  border : 0;
}
.e-control ul {
  border: 1px solid #ddd;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
.e-control .e-file-select,
.e-control .e-file-drop {
  display: none;
}
.e-upload.e-lib.e-keyboard {
  margin-top: 20px;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/invisible-cs2" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Customize progressbar in Vue Uploader component

You can customize the progress bar by override the styles in uploader component. In the following example, showcase about how to customize the progress bar's size, color and progress background.

APP.VUE

```
<template>
  <div id='upload_container' class='custom_progress'>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path"></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin, SelectedEventArgs } from '@syncfusion/ej2-vue-
inputs';
Vue.use(UploaderPlugin);
export default {
  data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
```

```
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .custom_progress .e-upload .e-upload-files .e-upload-file-list .e-file-
container .e-progress-inner-wrap .e-upload-progress-bar.e-upload-progress {
      background: yellow;
      height: 4px;
    }
     .custom_progress .e-upload .e-upload-files .e-upload-file-list .e-file-
container .e-progress-inner-wrap {
      height: 4px;
      background-color: lightblue;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/invisible-cs6" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Sort the selected files in Vue Uploader component

You can sort the selected files list and shown the list items in meaningful order. By using selected event, you can achieve this behavior in uploader component.

In the following example, explains about how to sort the file list in uploader component.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:selected = "onSelect" :asyncSettings= "path"></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin, SelectedEventArgs } from '@syncfusion/ej2-vue-
inputs';
Vue.use(UploaderPlugin);
let initial: boolean = true;
export default {
  data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          }
        }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    },
    methods: {
 onSelect: function(args: SelectedEventArgs): void {
    if (initial) { initial = false; return; }
    args.isModified = true;
    let oldFiles: FileInfo[] = this.$refs.uploadObj.getFilesData();
    let filesData: FileInfo[] = args.filesData.concat(oldFiles);
    let modifiedData: FileInfo[] = this.sortFileList(filesData);
    args.modifiedFilesData = modifiedData;
},
 sortFileList: function(filesData: FileInfo[]): FileInfo[] {
    let files: FileInfo[] = filesData;
    let fileNames: string[] = [];
    for (let i: number = 0; i < files.length; i++) {
        fileNames.push(files[i].name);
    }
    let sortedFileNames: string[] = fileNames.sort();
    let sortedFilesData: FileInfo[] = [];
    let index: number = 0;
    for (let name of sortedFileNames) {
        for (let i: number = 0; i < files.length; i++) {
            if (name === files[i].name) {
                sortedFilesData.push(files[i]);
            }
        }
    }
    return sortedFilesData;
}
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/invisible-cs9" %}

Copyright © 2001 -2024 Syncfusion Inc.

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Get the total size of selected files in Vue Uploader component

You can get the total size of selected files before upload it to designated server. By using selected event you can achieve this behavior.

In the following example, explains about how to calculate total file size before upload.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:selected = "onSelect" :asyncSettings= "path"></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin, SelectedEventArgs } from '@syncfusion/ej2-vue-
inputs';
Vue.use(UploaderPlugin);
export default {
  data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          }
        }
    },
    methods: {
  onSelect:function(args: SelectedEventArgs): void {
    let totalSize: number = 0;
    for (let file of args.filesData) {
        totalSize = totalSize + file.size;
    }
    let size: string = this.$refs.uploadObj.bytesToSize(totalSize);
    alert("Total select file's size is " + size);
}
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/invisible-cs8" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Customize button with html element in Vue Uploader component

The uploader component allows you to customize the action buttons by using buttons property.

In the following example explains about how to customize the action buttons.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:buttons="buttons" :autoUpload="autoUpload" :asyncSettings= "path"></ejs-
uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin, SelectedEventArgs } from '@syncfusion/ej2-vue-
inputs';
import { createElement } from '@syncfusion/ej2-base';
Vue.use(UploaderPlugin);
let uploadEle = createElement('span', { className: 'upload e-icons' });
uploadEle.innerHTML = 'Upload All';
let clearEle = createElement('span', { className: 'remove e-icons' });
clearEle.innerHTML = 'Clear All';
export default {
  data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
        autoUpload: false,
        buttons: {
        browse: 'Choose file',
        clear: clearEle,
        upload: uploadEle
        }
        }
```

```
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
.e-upload .e-upload-actions .e-file-clear-btn,
.e-upload .e-upload-actions .e-file-upload-btn {
  -webkit-tap-highlight-color: transparent;
    background-color: #ddd;
    border-color: #c3c3c3;
    color: #000000;
    box-shadow: none;
    width: 50%;
    margin: 0;
    text-transform: none;
    padding: 8px 0;
}
.upload::before {
  content: '\e60f';
  position: relative;
  font-size: 10px;
  right: 10px;
}
.remove::before {
  content: '\e932';
  position: relative;
  font-size: 10px;
  right: 10px;
}
.e-upload .e-upload-actions {
  width: 100%;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/invisible-cs5" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Copyright © 2001 -2024 Syncfusion Inc.

Add confirm dialog to remove the files in Vue Uploader component

You can customize the uploader component for use confirm dialog before remove the files. In the following example, used ej2 dialog as confirm dialog which is used for making confirmation on removing the files by using remove method.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload'
:removing="onremove" name="UploadFiles"  :asyncSettings= "path"></ejs-
uploader>
    <div id="dlgContainer">
     <ejs-dialog ref="dialogObj" :buttons="dlgButtons"
:target="dlgContainer" :visible="false" :width='width'
:content='dialogContent'>
     </ejs-dialog>
    </div>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin, SelectedEventArgs } from '@syncfusion/ej2-vue-
inputs';
import { createElement } from '@syncfusion/ej2-base';
import { DialogPlugin } from '@syncfusion/ej2-vue-popups';
Vue.use(DialogPlugin);
Vue.use(UploaderPlugin);
let removeFile: FileInfo[];
export default {
  data: function(){
        return {
          dlgContainer: "#dlgContainer",
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
            allowedExtensions: 'image/*',
            dialogContent: 'Confirm to remove the file?',
            width: '250px',
            dlgButtons: [{ click: this.dlgButtonClick, buttonModel: {
content: 'OK', isPrimary: true } },
            { click: this.cancelClick, buttonModel: { content: 'Cancel' }}]
        }
    },
    methods: {
        onremove:function(args: SelectedEventArgs): void {
            this.removeFile=[];
            args.cancel = true;
            this.removeFile.push(args.filesData);
            this.$refs.dialogObj.show();
        },
        dlgButtonClick: function() {
            this.$refs.dialogObj.hide();
            this.$refs.uploadObj.remove(this.removeFile[0], false, true);
```

```
            this.removeFile = [];
        },
        cancelClick: function(){
            this.$refs.dialogObj.hide()
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    #dlgContainer {
        min-height: 350px;
        width: 100%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/confirm-dialog-cs1" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Add additional data on upload in Vue Uploader component

The uploader component allows you to add the additional data on file upload which is used to get in the server end. By using uploading event and its customFormData argument, you can achieve this behavior.

In the following code snippet, explains about how to add additional data on file upload.

`

<template>

<ejs-uploader ref="uploadObj" id='defaultfileupload' :uploading="onFileUpload" name="UploadFiles" :autoUpload="autoUpload" :asyncSettings= "path"></ejs-uploader>

</template>

Copyright © 2001 -2024 Syncfusion Inc.

```
<script>
import Vue from 'vue';
import { UploaderPlugin, SelectedEventArgs } from '@syncfusion/ej2-vue-inputs';
import { createElement } from '@syncfusion/ej2-base';
Vue.use(UploaderPlugin);
export default {
data: function(){
return {
path:  {
saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
},
autoUpload: false,
allowedExtensions: 'image/*'
}
},
methods: {
onFileUpload:function(args) {
// add addition data as key-value pair.
args.customFormData = [{'name': 'Syncfusion INC'}];
}
}
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
container {
visibility: hidden;
padding-left: 5%;
width: 100%;
}
```

loader {
color: #008cff;

font-family: 'Helvetica Neue','calibiri';

font-size: 14px;

height: 40px;

left: 45%;

position: absolute;

top: 45%;

width: 30%;

}

</style>

`

Server side for adding additional data

`c#

// Get the additional data in server end by corresponding key.

var data = HttpContext.Current.Request.Form["name"];

`

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Validate image on drop in Vue Uploader component

The uploader component allows you to upload all type of images by setting image/* to allowedExtensions property or directly you can set it to accept attribute of uploader element.

By default, the behavior is working with select a file using browse button. But, this behavior doesn't support on drag and drop the files. You can handle this behavior manually using selected event by filtering the file types from application.

In the following example, validated image files using images/*. You are able to drag and drop the image files with extension of png, jpg, bpg, gif and tiff to upload it.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload'
:selected="onSelect" name="UploadFiles" :autoUpload="autoUpload"
:asyncSettings= "path" :allowedExtensions="extensions"></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin, SelectedEventArgs } from '@syncfusion/ej2-vue-
inputs';
```

```
import { DialogPlugin } from '@syncfusion/ej2-vue-popups';
import { createElement, detach } from '@syncfusion/ej2-base';
Vue.use(UploaderPlugin);
Vue.use(DialogPlugin);
export default {
  data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
            autoUpload: false,
            extensions: 'image/*'
        }
    },
    methods: {
          onSelect:function(args: SelectedEventArgs): void {
               if (args.event.type === 'drop') {
            let allImages: Array<string> = ['png', 'jpg', 'jpeg', 'gif',
'tiff', 'bpg'];
            let files = args.filesData;
            let modifiedFiles = [];
            for (let file of files) {
            if (allImages.indexOf(file.type) === -1) {
                file.status = 'File type is not allowed';
                file.statusCode = '0';
            }
            modifiedFiles.push(file);
            }
            args.isModified = true;
        }
        }
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/invisible-cs11" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Determine whether uploader has file input in Vue Uploader component

By setting **required** attribute to uploader input element, you can validate the file input has any value in it. In the below sample, set required attribute to the uploader input element and showcase the validation failure message using data-required-message attribute.

APP.VUE

```
<template>
        <div class="col-lg-12 control-section">
            <h4 class="form-title">Photo Contest</h4>
        <div class="control_wrapper" id ="control_wrapper">
            <form id="form1" method="post">
                <div class="form-group" style="padding-top: 40px; float:
left">
                    <div class="e-float-input">
                        <input type="text" id="name" name="name" data-
required-message="* Enter your name" required="" data-msg-
containerid="nameError">
                        <span class="e-float-line"></span>
                        <label class="e-float-text e-label-top"
for="name">Name</label>
                    </div>
                    <div id="nameError"></div>
                </div>
                <div id="dropArea">
                    <div id="uploadError" style='float: right;'></div>
                    <div id='customBrowse' class="form-group dropUpload">
Drop image here...
                    <ejs-uploader ref="uploadObj" id='defaultfileupload'
name="UploadFiles" :selected="onFileSelect" :autoUpload="autoUpload"></ejs-
uploader>
                    </div>
                    </div>
                <div class="submitBtn">
                    <button type="button" class="submit-btn e-btn"
id="submit-btn">Submit</button>
                    <div class="desc"><span>*This button is not a submit
type and the form submit handled from externally.</span></div>
                    <ejs-dialog id='uploadAlert' :header='header'
showCloseIcon=true :width= 'width' :content= 'dlgContent' :target=
'dlgTarget'
                :isModal= 'Modal' :visible="false" :animationSettings=
'animation' ref="dialogObj" >
                </ejs-dialog>
                </div>
            </form>
        </div>
    </div>
</template>
```

```
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
import { FormValidator  } from '@syncfusion/ej2-inputs';
import { DialogPlugin } from '@syncfusion/ej2-vue-popups';
import { createElement, detach } from '@syncfusion/ej2-base';
Vue.use(UploaderPlugin);
Vue.use(DialogPlugin);
export default {
  data: function(){
        return {
          dropElement: '.control-fluid',
          extensions: '.jpg, .png',
          isAuto: false,
          formObj: '',
          header: 'Success',
          animation: {effect: 'zoom'},
          Modal: true,
          dlgTarget: '.control_wrapper',
          dlgContent: 'Your details have been updated successfully, Thank
you.',
          width: '335px',
            autoUpload: false
        }
    },
    mounted: function() {
          let rules;
           let options = {
        //Initialize the CustomPlacement.
            customPlacement: function(inputElement, errorElement) {
                inputElement = inputElement.closest('.form-
group').querySelector('.error');
                inputElement.parentElement.appendChild(errorElement);
            },
             rules = {
                    'Name': {
                        required: true
                    },
                    'upload': {
                        required: true
                    }
                }
            }
    this.formObj = new FormValidator('#form1', options);
let localObj = this;
document.getElementById('submit-btn').onclick =             function() {
            localObj.onFormSubmit();
        };
        document.getElementById('customBrowse').onclick = () => {
    document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
};
  this.$refs.uploadObj.$el.setAttribute('data-required-message', '* Choose
your image to upload');
   this.$refs.uploadObj.$el.setAttribute('required', '');
  this.$refs.uploadObj.$el.setAttribute('data-msg-containerid',
'uploadError');
```

Copyright © 2001 -2024 Syncfusion Inc.

```
    },
    methods:{
    customBrowse: function() {
        document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
    },
    onFileSelect: function(args: any): void {
    if (args.filesData.length > 0) {
        if (document.getElementsByClassName('upload-image').length > 0) {
            detach(document.getElementsByClassName('imgWrapper')[0]);
        }
        let imageTag = createElement('IMG', { className: 'upload-image',
attrs: { 'alt': 'Image' } });
let wrapper = createElement('span', { className: 'imgWrapper' });
        wrapper.appendChild(imageTag);
            let rootFile = document.getElementsByClassName('dropUpload')[0];
rootFile.insertBefore(wrapper, rootFile.firstChild);
        this.readURL(wrapper, args.filesData[0]);
    }
    args.cancel = true;
},
readURL: function(li, args) {
    let preview = li.querySelector('.upload-image');
    let file: File = args.rawFile;
    let reader: FileReader = new FileReader();
    reader.addEventListener('load', () => { preview.src = reader.result; },
false);
    if (file) { reader.readAsDataURL(file); }
},
onFormSubmit: function(){
let formStatus = this.formObj.validate();
if (formStatus) {
        this.formObj.element.reset();
        detach(document.getElementsByClassName('imgWrapper')[0]);
        this.$refs.dialogObj.show();
    }
}
}
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        visibility: hidden;
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
```

```
        width: 30%;
    }
    .address-field {
  resize: none;
}
input#fileupload {
    opacity: 0;
}
#dropArea .dropUpload .upload-image {
  height: 140px;
  width: 140px;
}
#dropArea .dropUpload {
  float: right;
  text-align: center;
  vertical-align: middle;
  line-height: 12;
  overflow: hidden;
  border: 1px dashed;
  width: 150px;
  height: 150px;
}
.e-upload {
  visibility: hidden;
}
#control_wrapper {
  max-width: 500px;
  margin: auto;
  border: 0.5px solid #BEBEBE;
  box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.36);
  padding: 1% 4% 7%;
  background: #f9f9f9;
}
.e-error {
  padding-top:3px;
}
.control_wrapper .e-upload .e-upload-drag-hover {
  margin: 0;
}
.submit-btn {
  margin-top: 145px;
}
.submitBtn .desc {
  margin: 2% 23% 0 18%;
}
.submitBtn {
  text-align: center;
}
.form-support {
  width: 100%;
}
.success .form-support {
  display: none;
}
.success .successmsg {
  border: 0.5px solid green;
  padding: 10%;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
  color: green;
}
#form1 {
  position: relative;
  top: 14%;
}
.form-support td {
  width: 100%;
  padding-top:4%;
}
.e-upload {
  float: none;
}
.choose-file{
  width: 60%;
}
#browse {
  float: right;
  margin-right: -113px;
  margin-top: -27px;
}
.form-title {
  text-align: center;
}
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/invisible-cs7" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Achieve file upload programmatically in Vue Uploader component

You can upload a file programmatically using upload method. The selected files data, get from getFilesData public method in uploader.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:autoUpload="autoUpload" :asyncSettings= "path"></ejs-uploader>
    <div id="btnContainer">
        <span style='padding-left: 40px;'>
        <button id='first' style='margin-top: 30px' class='e-btn e-
control'>Upload first file</button>
    </span>
    <span style=' padding-left: 40px;'>
        <button style='margin-top: 30px' id='full' class='e-btn e-
control'>Upload all files</button>
    </span>
  </div>
  </div>
</template>
<script>
import Vue from 'vue';
```

Copyright © 2001 -2024 Syncfusion Inc.

```
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
  data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
            autoUpload: false
        }
    },
    mounted: function() {
       document.getElementById('first').onclick = (args) => {
    this.$refs.uploadObj.upload(this.$refs.uploadObj.getFilesData()[0]);
};
document.getElementById('full').onclick = (args) => {
    this.$refs.uploadObj.upload(this.$refs.uploadObj.getFilesData());
};
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .e-upload-actions {
        display: none;
    }
    .e-btn {
        text-transform: none;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/invisible-cs1" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Copyright © 2001 -2024 Syncfusion Inc.

Check file size before uploading it in Vue Uploader component

By using uploading event, you can get the file size before upload it to server. File object contains the file size in bytes only. You can convert the size to standard formats (KB or MB) using bytesToSize method.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:autoUpload="autoUpload" :asyncSettings= "path"
:uploading="onBeforeUpload"></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
  data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
            autoUpload: false
        }
    },
    methods: {
onBeforeUpload: function(args): void {
    // get the file size in bytes
    let sizeInBytes: number = args.fileData.size;
    // get the file size in standard format
    alert("File size is: " + this.$refs.uploadObj.bytesToSize(sizeInBytes))
}
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
```

Copyright © 2001 -2024 Syncfusion Inc.

```
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/invisible-cs3" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Check the mime type of file before upload it in Vue Uploader component

By using uploading event, you can get the file MIME type before uploading it to server. In the below sample, file MIME type is shown in the alert box before file start to upload.

APP.VUE

```
<template>
  <div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:autoUpload="autoUpload" :asyncSettings= "path"
:uploading="onBeforeUpload"></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
  data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          },
            autoUpload: false
        }
    },
    methods: {
onBeforeUpload: function(args): void {
 // get the file MIME type
    alert("File MIME type is: " + args.fileData.rawFile.type)
}
    }
}
</script>
<style>
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
```

```
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/invisible-cs4" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Trigger click event of input file from external button in Vue Uploader component

You can trigger the input type file click event from external button. In the below sample, click event of input file was triggered from Essential JavaScript 2 Button.

APP.VUE

```
<template>
  <div id="uploaderContainer">
    <!-- Initialize Uploader -->
<div id="dropArea">
    <span id="drop"> Drop image (JPG, PNG) files here or <button class='e-
btn e-control' id="browse">Browse</button></span>
</div>
    <ejs-uploader ref="uploadObj" id='defaultfileupload' name="UploadFiles"
:asyncSettings= "path"></ejs-uploader>
  </div>
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';
Vue.use(UploaderPlugin);
export default {
  data: function(){
        return {
          path:  {
            saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
            removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
          }
        }
    },
    mounted:function {
    document.getElementById('browse').onclick = () => {
        document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
    };
    }
}
</script>
<style>
```

```
@import "../../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
 #container {
        padding-left: 5%;
        width: 100%;
    }
    #loader {
        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size: 14px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    #dropArea {
        border: 1px dashed #c3c3cc;
        text-align: center;
        padding: 20px 0 10px;
    }
    .e-file-select-wrap {
        display: none;
    }
    .e-upload.e-lib.e-keyboard {
      border-bottom: 0px;
      border-top: 0px
    }
</style>
```

{% previewsample "page.domainurl/code-snippet/uploader/invisible-cs10" %}

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Open and edit the uploaded files in Vue Uploader component

The uploader component allows you to modify the file after uploading to the server, which can be achieved using success event of the uploader.

You can retrieve the saved file path in the uploader success event and assign it to custom attribute (data-file-name) value of the respective file list element to open the uploaded file. Click the respective file element to create a new request along with saved file path using http header. In the server-side, get the file path from the header and open the file using process.start method

`

<template>

<ejs-uploader ref="uploadObj" id='defaultfileupload' :success="onUploadSuccess" name="UploadFiles" :asyncSettings= "path"></ejs-uploader>

Copyright © 2001 -2024 Syncfusion Inc.

```
</template>
<script>
import Vue from 'vue';
import { UploaderPlugin, SelectedEventArgs } from '@syncfusion/ej2-vue-inputs';
import { createElement } from '@syncfusion/ej2-base';
Vue.use(UploaderPlugin);
export default {
data: function(){
return {
path:  {
saveUrl: 'YourHosedService/Home/Save',
removeUrl: 'YourHosedService/Home/Remove'
}
}
},
methods: {
onUploadSuccess: function(args) {
let liElements = document.body.querySelectorAll('.e-upload-file-list');
for (let i = 0; i < liElements.length; i++) {
if (liElements[i].getAttribute('data-file-name') == args.file.name) {
liElements[i].addEventListener('click', () => { this.openFile(args, event); });
// File path have to update from server end in response status description.
liElements[i].setAttribute('file-path', args.e.target.statusText);
}
}
},
openFile: function(args, e) {
if (!e.target.classList.contains('e-file-delete-btn') && !e.target.classList.contains('e-file-remove-btn'))
{
let ajax = new XMLHttpRequest();
// create new request for open the selected file
ajax.open("POST", "YourHosedService/Home/openFile');
let liElements = document.getElementsByClassName('e-upload')[0].querySelectorAll('.e-upload-file-list');
```

```
for (let i = 0; i < liElements.length; i++) {
if (liElements[i].getAttribute('data-file-name') == args.file.name) {
// Added the file path in header to get it in server side.
ajax.setRequestHeader('filePath', liElements[i].getAttribute('file-path').toString());
}
}
ajax.send();
}
}
}
}
</script>
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
container {
visibility: hidden;
padding-left: 5%;
width: 100%;
}
loader {
color: #008cff;
font-family: 'Helvetica Neue','calibiri';
font-size: 14px;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
</style>
`
```

Copyright © 2001 -2024 Syncfusion Inc.

Server side for open and edit the uploaded files

```c#
public void Save()
{
var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
var currentPath = System.Web.HttpContext.Current.Server;
var fileSave = currentPath.MapPath("UploadedFiles");
var fileSavePath = System.IO.Path.Combine(fileSave, httpPostedFile.FileName);
if (!System.IO.File.Exists(fileSavePath))
{
httpPostedFile.SaveAs(fileSavePath);
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusDescription = fileSavePath;
Response.End();
}
}
[AcceptVerbs("Post")]
public void openFile()
{
// Check whether the file is available in the corresponding location
if (System.IO.File.Exists(Request.Headers.GetValues("filePath").First()))
{
// This will open the selected file from server location in desktop
Process.Start(Request.Headers.GetValues("filePath").First());
}
}
```

Add the `filePath` in allow-headers in `web.config` file.

```c#
<customHeaders>
<add name="Access-Control-Allow-Headers" value="accept, maxdataserviceversion, origin, x-requested-with, dataserviceversion,content-type, chunk-index, Authorization, filePath" />
```

</customHeaders>

`

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Resize images before uploading it to the server in Vue Uploader component

You can customize the dimension of the images before uploading it to the server. By using selected event, you can get the selected file information as type of an object. From the obtained image file information, create a new canvas and render an image with the custom dimensions. Refer the corresponding code snippet as follows.

`

<template>

Drop files here or <u>Browse</u>

<ejs-uploader id='template' :multiple='false' name="UploadFiles" :asyncSettings= "path" ref="uploadObj"

:dropArea= "dropElement" :selected= "onFileSelect" :progress= "onFileUpload"

:success= "onUploadSuccess" :failure= "onUploadFailed" :removing= "onFileRemove">

</ejs-uploader>

</template>

`

`ts

<script>

import Vue from "vue";

import { UploaderPlugin } from '@syncfusion/ej2-vue-inputs';

import { FileInfo } from '@syncfusion/ej2-vue-inputs/uploader';

import { createSpinner, showSpinner, hideSpinner } from '@syncfusion/ej2-popups';

import { createElement, isNullOrUndefined, detach, EventHandler } from '@syncfusion/ej2-base';

Vue.use(UploaderPlugin);

export default Vue.extend({

data: function() {

return {

```
path:  {
saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
},
dropElement: '#dropElement',
filesList: [],
filesDetails: [],
parentElement: '',
progressbarContainer: ''
}
},
mounted: function () {
document.getElementById('browse').onclick = function() {
document.getElementsByClassName('e-file-select-wrap')[0].querySelector('button').click();
return false;
};
},
methods:{
onFileSelect: function (args) {
args.cancel = true;
if (isNullOrUndefined(document.getElementById('dropTarget').querySelector('.upload-list-root'))) {
this.parentElement = createElement('div', { className: 'upload-list-root' });
this.parentElement.appendChild(createElement('ul', {className: 'ul-element' }));
document.getElementById('dropTarget').appendChild(this.parentElement);
}
for (let i = 0; i < args.filesData.length; i++) {
this.formSelectedData(args.filesData[i]);
}
this.filesDetails = this.filesDetails.concat(args.filesData);
let file = args.filesData[0].rawFile;
let width;
let height;
let img = document.createElement("img");
```

Copyright © 2001 -2024 Syncfusion Inc.

```
let reader = new FileReader();
reader.onload = function(e) { img.src = e.target.result; };
reader.readAsDataURL(file);
let imgs = new Image();
let localObj = this;
img.onload = function() {
console.log(this);
width = this.width;
height = this.height;
localObj.onNewImg(height, width, img, args.filesData[0]);
},
imgs.src = img.src;
},
// to create canvas and update our custom dimensions
onNewImg(height, width, img, file) {
let canvas = document.createElement("canvas");
let ctx = canvas.getContext("2d");
ctx.drawImage(img, 0, 0);
let MAX_WIDTH = 1000;
let MAX_HEIGHT = 600;
if (width > height) {
if (width > MAX_WIDTH) {
height *= MAX_WIDTH / width;
width = MAX_WIDTH;
}
} else {
if (height > MAX_HEIGHT) {
width *= MAX_HEIGHT / height;
height = MAX_HEIGHT;
}
}
canvas.width = width;
canvas.height = height;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
let ctx1 = canvas.getContext("2d");
ctx1.drawImage(img, 0, 0, width, height);
let newImage = canvas.toDataURL("image/png");
let blobBin = atob(newImage.split(',')[1]);
let array = [];
for(var i = 0; i < blobBin.length; i++) {
array.push(blobBin.charCodeAt(i));
}
let newBlob = new Blob([new Uint8Array(array)], {type: 'image/png'});
let newFile = this.createFile(newBlob, file);
this.$refs.uploadObj.upload(newFile, true);
},
// To create File object to upload
createFile(image, file) {
let newFile = {
name: file.name,
rawFile: image,
size: image.size,
type: file.type,
validationMessage: '',
statusCode: '1',
status: 'Ready to Upload'
}
return newFile;
},
formSelectedData: function ( selectedFiles) {
let liEle = createElement('li',  { className: 'file-lists', attrs: {'data-file-name' : selectedFiles.name} });
liEle.appendChild(createElement('span', {className: 'file-name ', innerHTML: selectedFiles.name }));
liEle.appendChild(createElement('span', {className: 'file-size ', innerHTML:
this.$refs.uploadObj.bytesToSize(selectedFiles.size) }));
if (selectedFiles.statusCode === '1') {
this.progressbarContainer = createElement('span', {className: 'progress-bar-container'});
this.progressbarContainer.appendChild(createElement('progress', {className: 'progress', attrs: {value :
'0', max : '100'}} ));
```

```
liEle.appendChild(this.progressbarContainer);
} else { liEle.querySelector('.file-name').classList.add('upload-fails'); }
let closeIconContainer = createElement('span', {className: 'e-icons close-icon-container'});
let localObj = this;
closeIconContainer.addEventListener( 'click', function(e) {
localObj.removeFiles(e);
});
liEle.appendChild(closeIconContainer); document.querySelector('.ul-element').appendChild(liEle);
this.filesList.push(liEle);
},
onFileUpload: function(args) {
let li = document.getElementById('dropTarget').querySelector('[data-file-name="' + args.file.name + '"]');
let localObj = this;
let progressValue = Math.round((args.e.loaded / args.e.total) * 100);
if (!isNaN(progressValue)) {
li.getElementsByTagName('progress')[0].value = progressValue;
}
},
onUploadSuccess: function(args) {
let spinnerElement = document.getElementById('dropTarget');
let li= document.getElementById('dropTarget').querySelector('[data-file-name="' + args.file.name + '"]');
if (!isNullOrUndefined(li.querySelector('.progress-bar-container'))) {
detach(li.querySelector('.progress-bar-container'));
}
let localObj = this;
if (args.operation === 'upload') {
li.querySelector('.file-name').classList.add('upload-success');
li.querySelector('.close-icon-container').classList.remove('remove-btn');
li.querySelector('.close-icon-container').classList.add('delete-icon');
(li.querySelector('.close-icon-container')).onclick = function() {
localObj.generateSpinner(li.querySelector('.close-icon-container'));
};
(li.querySelector('.close-icon-container')).onkeydown = function(e) {
```

Copyright © 2001 -2024 Syncfusion Inc.

```
if (e.keyCode === 13) {
localObj.generateSpinner(e.target.closest('.e-upload'));
}
};
}
if (args.operation === 'remove') {
this.filesDetails.splice(this.filesList.indexOf(li), 1);
this.filesList.splice(this.filesList.indexOf(li), 1);
detach(li);
hideSpinner(li.querySelector('.close-icon-container'));
detach(li.querySelector('.e-spinner-pane'));
}
alert(args.file.size);
},
generateSpinner: function(targetElement) {
createSpinner({ target: targetElement, width: '25px' });
showSpinner(targetElement);
},
onUploadFailed: function(args) {
let li = document.getElementById('dropTarget').querySelector('[data-file-name="' + args.file.name + '"]');
let localObj = this;
li.querySelector('.file-name').classList.add('upload-fails');
li.querySelector('.close-icon-container').classList.remove('remove-btn');
if (args.operation === 'remove') {
if (!isNullOrUndefined(li)) {
this.filesDetails.splice(this.filesList.indexOf(li), 1);
this.filesList.splice(this.filesList.indexOf(li), 1);
detach(li);
}
}
if (args.operation === 'upload') {
detach(li.querySelector('.progress-bar-container'));
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
},
removeFiles: function(args) {
if (!isNullOrUndefined(args.currentTarget)) {
if (this.filesDetails[this.filesList.indexOf(args.currentTarget.parentElement)].statusCode === '3') { return;}
if (this.filesDetails[this.filesList.indexOf(args.currentTarget.parentElement)].statusCode === '2' ) {
this.$refs.uploadObj.remove(this.filesDetails[this.filesList.indexOf(args.currentTarget.parentElement)]);
} else  {
this.onFileRemove(args);
}
}
},
onFileRemove: function(args) {
args.postRawFile = false;
if (!isNullOrUndefined(args.currentTarget)) {
if (this.filesDetails[this.filesList.indexOf(args.currentTarget.parentElement)].statusCode !== '2') {
detach(args.currentTarget.parentElement);
this.filesList.splice(this.filesList.indexOf(args.currentTarget.parentElement), 1);
}
}
}
}
});
</script>
`
`
<style>
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-vue-inputs/styles/material.css";
dropTarget {
min-height: 50px;
padding-top: 15px;
position: relative;
```

```
}
dropElement {
padding: 3% 30% 3%;
display: inherit;
border: 1px dashed #c3c3cc
}
.droparea {
font-size: 13px;
}
.e-file-select-wrap {
display: none;
}
.e-upload {
float: none;
border: none;
}
.ul-element {
list-style: none;
width: 100%;
padding-left: 0;
}
.file-name {
padding: 8px 6px 8px 0;
font-size: 13px;
width: 46%;
display: inline-block;
position: relative;
top: 4px;
}
.file-size {
padding: 4px;
font-size: 13px;
width: 18%;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
display: inline-block;
position: relative;
}
.file-lists {
border: 1px solid lightgray;
padding: 0 6px 0 14px;
margin-top: 15px;
position: relative;
background: rgba(0, 0, 0, 0.04);
}
.file-size, .file-name {
font-family: "Helvetica Neue", "Helvetica", "Arial", "sans-serif";
text-overflow: ellipsis;
overflow: hidden;
white-space: nowrap;
}
span.progress-bar-container {
display: block;
float: right;
height: 20px;
right: 13%;
top: 14px;
position: relative;
width: 20%;
}
.progress{
width: 100%;
height: 15px;
-webkit-appearance: none;
}
.close-icon-container
{
cursor: pointer;
```

Copyright © 2001 -2024 Syncfusion Inc.

```
font-size: 11px;
height: 24px;
margin: 0 12px 0 22px;
padding: 0;
position: absolute;
right: 0;
width: 24px;
top: 6px;
}
.close-icon-container.remove-btn {
display: none;
}
.close-icon-container.e-icons::before {
left: 7px;
position: inherit;
top: 7px;
content: '\e932';
}
.close-icon-container.delete-icon::before {
content: '\e94a';
}
.close-icon-container:hover {
background-color: rgba(0, 0, 0, 0.12);
border-color: transparent;
border-radius: 50%;
box-shadow: 0 0 0 transparent;
}
.upload-success {
color: #2bc700;
}
.upload-fails {
color: #f44336;
}
```

Copyright © 2001 -2024 Syncfusion Inc.

```
progress::-webkit-progress-bar {
border: 1px solid lightgrey;
background-color: #ffffff;
border-radius: 2px;
}
dropTarget progress {
border: 1px solid lightgrey;
background-color: #ffffff;
border-radius: 2px;
}
progress::-webkit-progress-value {
border-radius: 2px;
background-color: #ff4081;
}
dropTarget span a {
color:#ff4081;
}
</style>
`
```

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Convert image into binary format after uploading in Vue Uploader component

By default, the file upload component saves the uploaded image files in physical directories. Also, you can convert the images into binary format at server-side before saving the uploaded images. To retrieve binary format of image files, convert the posted file's input stream into binary reader and read as bytes using ReadBytes method.

Refer to the below server-side code snippet

```
`c#
[AcceptVerbs("Post")]
public void Save()
{
try
{
if (System.Web.HttpContext.Current.Request.Files.AllKeys.Length > 0)
```

```
{
var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
if (httpPostedFile != null)
{
byte[] fileBytes;
using (BinaryReader br = new BinaryReader(httpPostedFile.InputStream))
{
fileBytes = br.ReadBytes((int)httpPostedFile.InputStream.Length);
// bytes will be stored in variable fileBytes
}
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusCode = 200;
Response.Status = "200 Success";
Response.End();
}
}
}
catch (Exception e)
{
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusCode = 204;
Response.Status = "204 No Content";
Response.StatusDescription = e.Message;
Response.End();
}
}
`
```

You can also explore Vue File Upload feature tour page for its groundbreaking features. You can also explore our Vue File Upload example to understand how to browse the files which you want to upload to the server.

Copyright © 2001 -2024 Syncfusion Inc.

Copyright © 2001 -2024 Syncfusion Inc.